THE BEGINNER'S COMPUTER HANDBOOK

Understanding & programming the micro
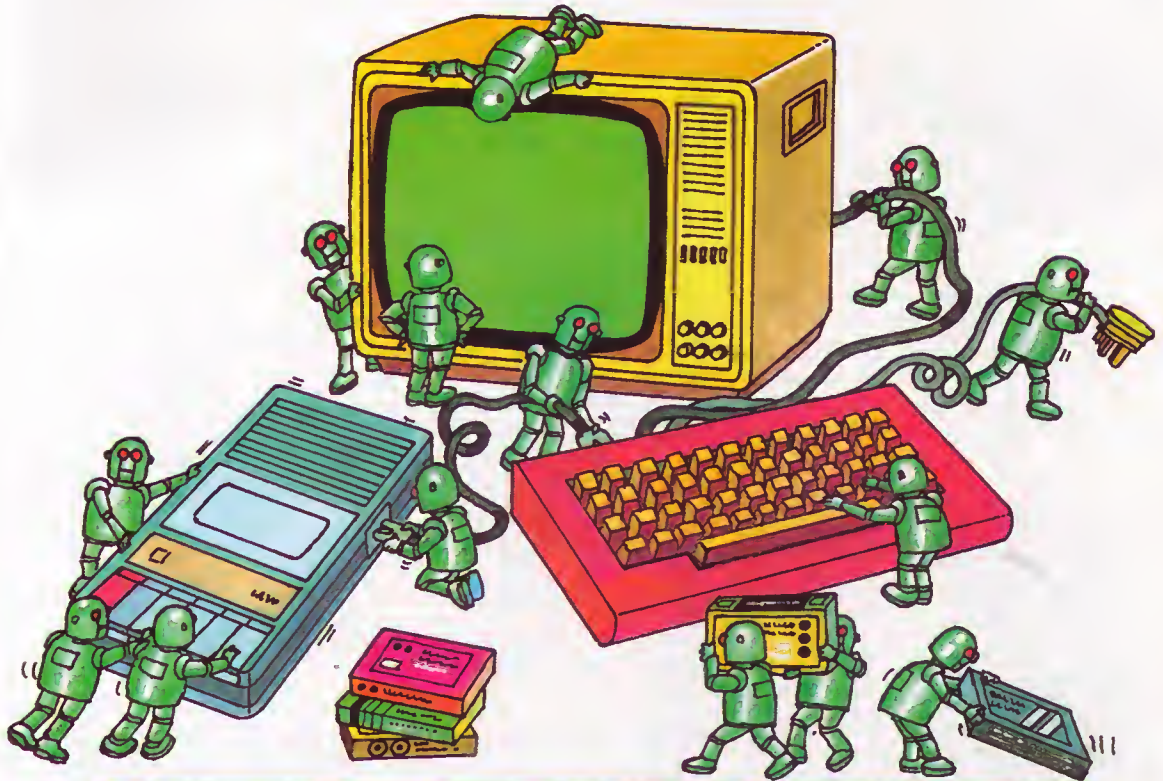
F 12·10

A COMPLETE GUIDE

USBORNE

UNDERSTANDING THE
MICRO

Judy Tatchell and Bill Bennett
## Edited by Lisa Watts

Contents

2

This part of the book is for anyone who wants to know about microcomputers. It shows what you can do with them, how you use them, and how they work. It explains computer jargon so you can go on to read and understand more about computers.
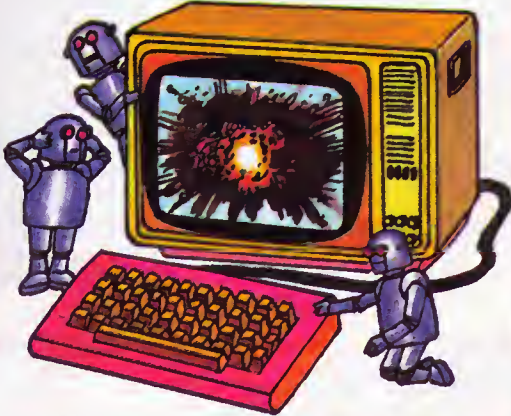
Microcomputers are small, multi-purpose computers. You can play games on them, draw pictures and sometimes even make sounds and music. They can also do complicated sums very quickly, and you can keep diaries and catalogues of records and slides, or anything else you collect.

The first part of the book explains how to use a micro and how you give it a program telling it what to do. There is an introduction to writing programs in BASIC, which is the programming language most micros understand, and there are lots of programming hints. If you have access to a micro, there are some games programs you can try out.
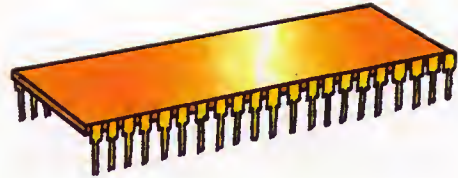
The book then describes how a micro works, and how it makes pictures and different sounds. It shows the inside of a micro with its tiny silicon chips which do all the processing. You can also find out how some micros can be linked to other computers thousands of miles away to bring all sorts of information into your own home. Micros can be used to control robots, or other electronic equipment, such as model railways, too.

Although to begin with you only need an ordinary television set to use with a micro, you can buy lots of other pieces of equipment to connect to it – a light pen for drawing pictures directly on the screen, for instance, or special attachments for using with arcade-type games. The book describes these, too.
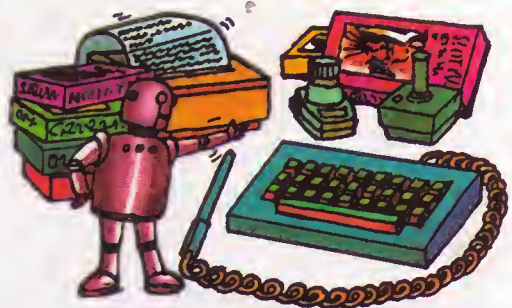
On page 40 there is a guide to buying a micro. It tells you about some of the most popular home computers so you can compare them and explains the terminology used to describe a computer.

3

Meet the micro

These two pages show a micro and how to set it up. Not all micros look exactly like the one in the picture. Most home computers, though, consist of a keyboard that you connect to a TV. Some micros have screens specially designed for them. These are called visual display units (VDUs), or monitors. All new micros are supplied with manuals to tell you how to use them. Before setting up a micro, check its manual for special instructions.

Screen

Keyboard
You give the micro instructions and information by typing on the keyboard.

Keyboard

Sockets where leads from the TV and the mains plug into the micro.

All the important parts of the micro are kept inside the keyboard, where it does its work.

Where the micro does its work

Input

The micro has an electronic memory where it stores information and instructions.

Output
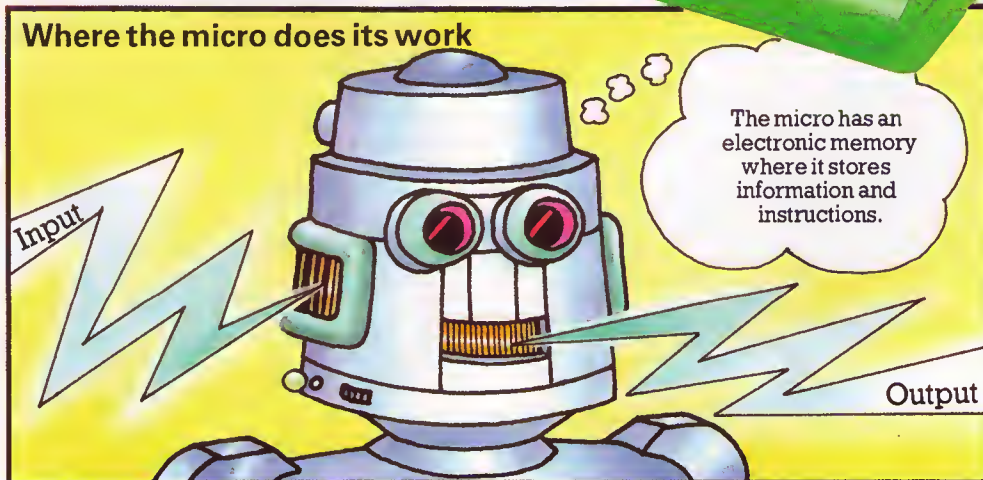
The "brain" of the micro is usually inside the keyboard. It consists of a central processing unit (CPU) which does all the work, and a memory. Before it can do anything the CPU needs a set of instructions called a program. This is stored in the memory along with the information, or data, you want it to work on. Programs and data are called input. The results are called output.

ZAP

Screen

Everything you type on the keyboard, and the results of the micro's work, appear on the screen. The micro can also draw pictures and make shapes on the screen. Most micros can make coloured pictures if they are connected to a colour TV.
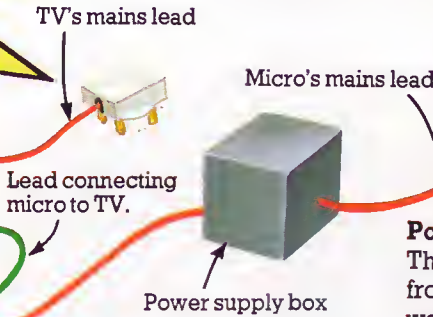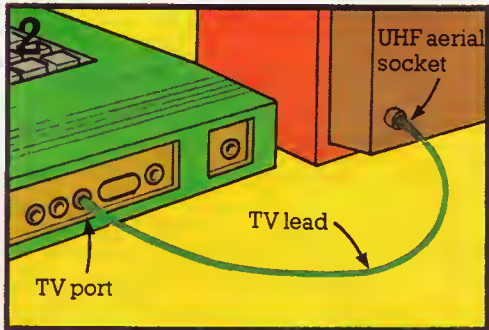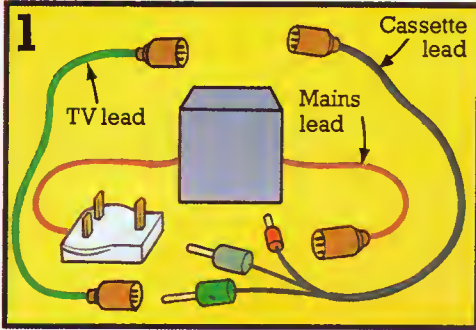
TV's mains lead

Micro's mains lead

Some micros can produce music and sound effects.

Lead connecting micro to TV.

Power supply box

Power supply

The power supply box reduces the power from the mains to a level the micro can work on, and keeps the power supply smooth.

How to set up a micro

1

Cassette lead

TV lead

Mains lead

Most micros have three leads, one to link the keyboard to a TV, one to plug the keyboard into the mains electricity supply, and a third to connect it to a cassette recorder.*

2

UHF aerial socket

TV lead

TV port

To connect the keyboard to a TV, pull the aerial plug out of the TV. Then plug one end of the micro's TV lead into the hole marked TV on the keyboard and the other into the UHF socket on the TV.

3

Mains lead
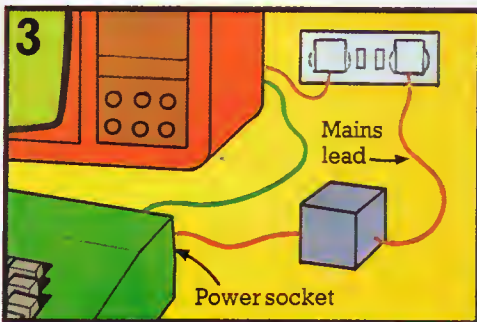
Power socket

Plug one end of the mains lead into the power socket on the keyboard and the other end into the mains at the wall. Make sure the TV is also plugged in, then switch them both on.

4

READY

Select a TV channel which you are not using for TV programmes. Tune the TV until the micro's "ready" signal appears on the screen. These signals vary from micro to micro.

5

*You use the cassette recorder to store programs for the micro. This is explained on page 16.

Programming a micro

Whether you want to use your micro to play a space game or simply to add some numbers together, you have to give it a program of instructions to tell it what to do. There are special computer languages for writing programs. They consist of words and symbols the computer can recognize and convert into its own electronic code, called machine code. Program instructions are stored in the computer's memory and then carried out by the CPU. Programs and data which you give the micro are called computer software. Parts of the micro that you can touch, like the keyboard and the screen, are called computer hardware.

1 Telling a computer what to do

1. Lean ladder against wall
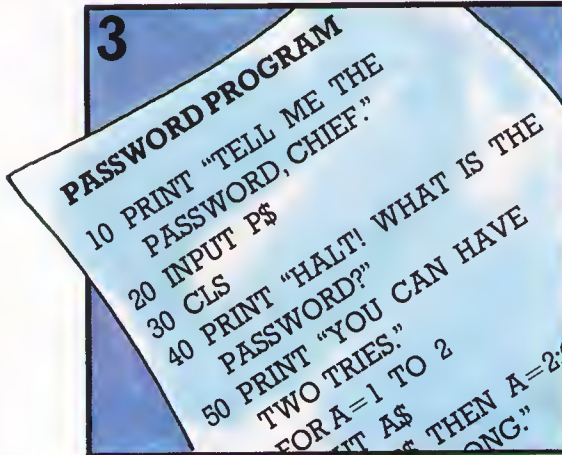2. Climb ladder
3. Paint window frame
4. Go down ladder
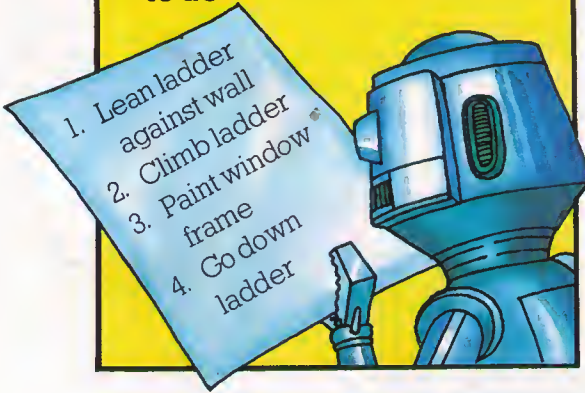
A computer can only carry out a task if it is told exactly what to do in the right order. This program tells a robot with a computer brain how to paint a window.

2

The program would not work as there is no instruction telling the robot to pick up the paint pot and brush before climbing the ladder. The robot only does what it is told to do.

3

PASSWORD PROGRAM

```
10 PRINT "TELL ME THE
       PASSWORD, CHIEF."
20 INPUT P$
30 CLS
40 PRINT "HALT! WHAT IS THE
       PASSWORD?"
50 PRINT "YOU CAN HAVE
       TWO TRIES."
   FOR A=1 TO 2
   ...T A$
   ... THEN A=2:
   ...NG."
```

This is part of a program* in BASIC, the language most micros use. A computer contains a set of instructions called an interpreter which translates the programming language into machine code.

4

> blip blip blop blip blop blop blop blip blop

> blip blop blip blip blop blop
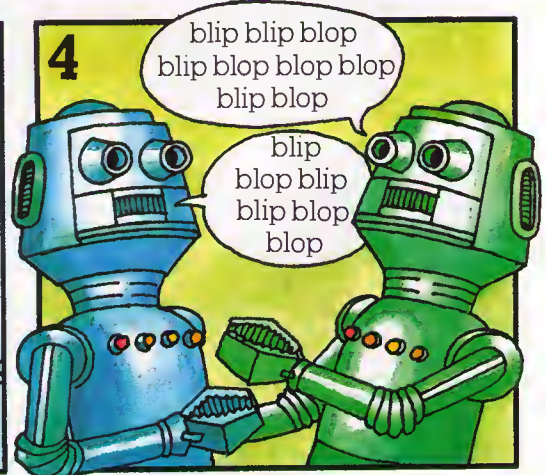
All the work inside the computer is done in machine code. Each "word" of the code consists of patterns of pulses in the electric current flowing round the computer.

6

*This program is printed out in full on page 12.

The computer's memory

A computer has two kinds of memory. One is a permanent store of instructions which tell it how to work. The other is an empty memory where your program and data for a job are stored temporarily. Each time the micro is switched off, the memory empties again.

The temporary memory is called RAM (random access memory). It is sometimes called a read/write memory. Everything you put into the micro is stored or "written" in here for you to "read" or refer to, and you can also change it.

The permanent memory is called ROM (read only memory). The name means that the micro can only take, or "read", information from it. You cannot store extra information there. The interpreter is stored in the ROM.

▲
ROM is like an instruction manual. The micro can only read from it and it cannot rub it out or store new information there.

▲
RAM is like a notepad. The micro can write in it as well as read from it. It is rubbed out whenever the micro is switched off.

Memory size
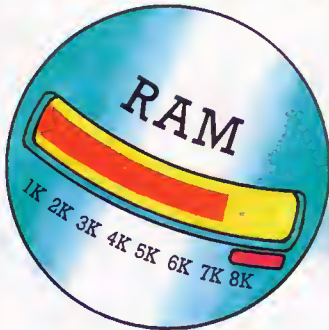
Micros come with different sized memories. Memory size is measured by the number of machine code "words" that can be stored. Each code word is called a byte and 1024 bytes are called a kilobyte, or 1K.

One kilobyte is about the same as 500 BASIC words or symbols. It is enough to store simple programs. More advanced programs are longer and might need 8K or 16K of RAM. You can buy extra RAM, called add-on RAM packs, for most micros.

Looking at the keyboard

The keyboard of a micro usually looks much like a typewriter keyboard. It has the same letters and numbers, arranged in the same order. A micro also has some extra keys, though, for giving special commands in BASIC. The micro receives different electrical messages from each key. If you type in something the micro does not recognize, a message telling you so will appear on the screen saying "Error" or "Mistake". Everything you type is stored in the micro's temporary memory (RAM), and also displayed on the screen for you to check. These two pages show two different keyboards.

Letter keys
On most computers you type in a program using the symbol keys and spelling out the words with the letter keys.

This is the computer's multiplication sign.

Shift key and shift lock

Other micros might have some different keys.
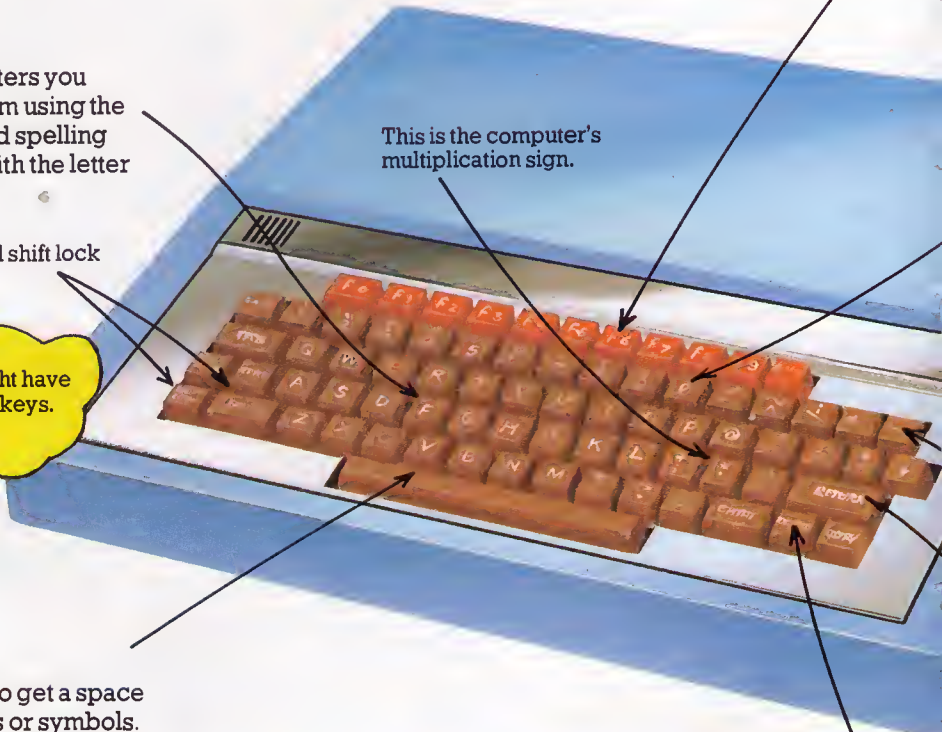
Space bar
You press this to get a space between words or symbols.

Using the shift key
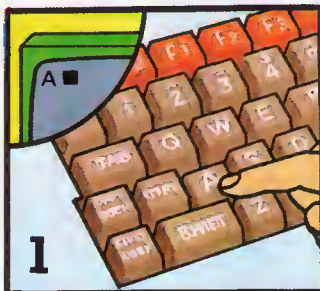
1
Most micros automatically make capital letters on the screen and cannot make small letters.
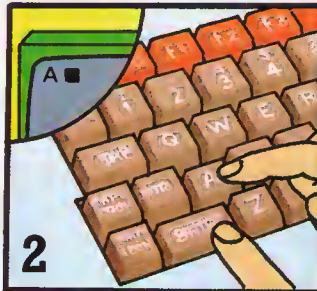
2
Some micros, though, make small letters. To make a capital letter, you hold the shift key down while you press the letter key.
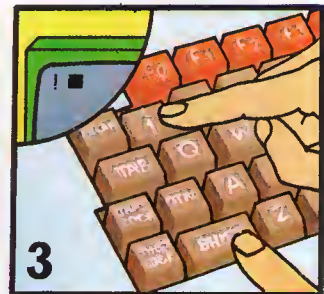
3
Where there are two symbols on the same key, you use the shift key to get the top one. Pressing a key without the shift key gives the lower one.

Programmable keys
These are special keys which you can program yourself to do special jobs such as producing certain colours each time they are pressed. Not all micros have these keys.

The figure zero on a computer usually has a stroke through it to distinguish it from the capital letter O.

1 The cursor

The cursor is a little marker that moves across the screen as you type, to show where the next letter will appear.

2

If you want to change or delete something, you can move the cursor back over your typing using the cursor control keys.
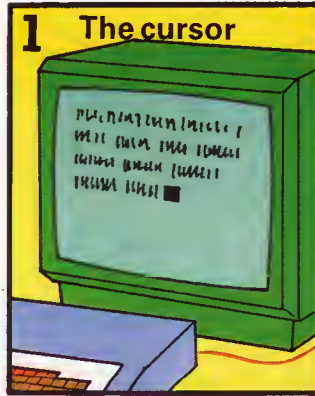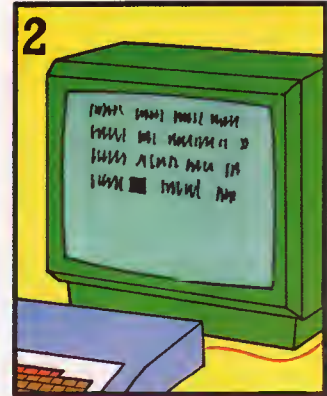
Another micro

This keyboard is about a quarter of the size of the keyboard on the left. The design of the typing area can determine the size and shape of a micro, as the parts inside are very small. Also, there is usually room on the keyboard for sockets into which you can plug extra things like a printer or a cassette recorder. These are described later in the book.

Cursor control keys

Return key
At the end of each line of the program you press this key to start a new line. It also enters the line you have just typed into the micro's memory. It is sometimes called NEWLINE or ENTER.

Delete key
You can rub out mistakes you have typed using this key. On some micros it is called RUBOUT or ERASE.

This kind of micro has calculator-like keys which do not move much when you press them. Most of the keys carry complete BASIC words so you do not have to spell them out letter by letter. The keys have words, letters and symbols on them, and there are two different shift keys for selecting which message you want from a key.

9

Programs for the micro

You can buy programs in magazines, books, recorded on cassette tape or disk – or you can learn to write your own. Programs printed out line by line are called listings. Programs on cassette can be loaded into a micro using a cassette recorder. The program must be written in the correct language for the micro. This is usually BASIC, but there are several different "dialects" with different commands. The program will not work if the dialect is wrong or if it has a mistake.

1 What you can buy

You can buy all kinds of games programs from arcade-type games with colour pictures and exciting sound effects to more traditional ones like chess.

Where to get programs

You can buy microcomputer magazines containing listings at most newsagents. Some are produced specifically for one kind of micro. Others have programs for several different micros.

Another quite cheap way of getting programs is to buy collections of them in books. These are usually games programs written for one particular micro.

Often the programs in magazines have not been thoroughly tested and contain mistakes, called "bugs", which stop the programs working. Programs in books are usually more reliable.

Typing in listings helps you get to know BASIC and you learn to spot bugs and adapt programs if the dialect is wrong for your micro.

Prestel TV

ROM cartridges

You can sometimes get programs displayed on your TV screen. Your TV needs to be a special one that can be linked by telephone to a Viewdata system, like Prestel. These are computerized information centres. You choose which page of information you want shown.

For some micros you can buy programs in cartridges like these. You plug the cartridge into the micro and the program is automatically loaded into the micro's memory.

10
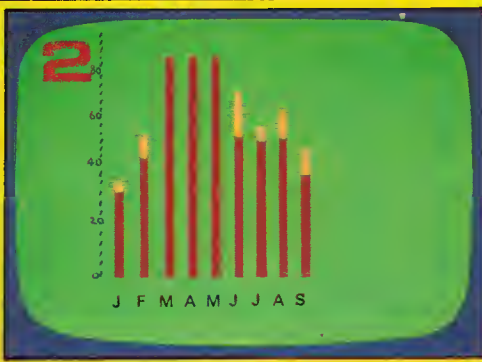
You can organize your home life with programs for keeping accounts and diaries, as well as catalogues for things you collect. These are simpler versions of programs used in business.

Educational programs can help with all kinds of learning, from spelling and maths to speaking a foreign language. Pictures on the screen often help make things clear and more interesting.

HINTS

1. Make sure the program is written in the dialect of BASIC understood by the micro you are using.

2. As a rough guide, 1K of memory is enough for about 40 program lines.

3. If a program on tape or disk that should be suitable for the micro does not work, write and tell the suppliers.

User groups

User groups provide an opportunity to meet other people interested in micros and to exchange programs and ideas. You can find out if there is one in your area by writing to a micro magazine, or look at the lists of societies at your local library.

Floppy disks

You can buy programs recorded on cassette, for which you need a cassette player. You can also buy them on floppy disk. These are made from the same material as cassette tapes, but are more expensive and you need a disk drive to use them. Shops and mail order companies sell cassettes and disks. You can find the names of suppliers in micro magazines and write off for catalogues.

Writing your own programs

Most micros are designed to understand BASIC, which is a good, general-purpose programming language. There are lots of other languages, though, and some people think Pascal is better. There are two programs in BASIC on this page. BASIC consists of symbols and words, and is quite easy to learn. The best way to start is to read lots of programs and it helps if you have a micro to try them out on. Most micros' manuals explain BASIC. You can also buy books on how to program, or you can get instruction courses on tape or disk to run on a micro.

> You can find out what some BASIC terms mean in the Password Program on the right. This program is to stop spies infiltrating a secret society...

Each line of the program is numbered. The numbers usually go up in tens so you can insert extra lines into the program later, if necessary, without having to renumber all of them. The micro follows the program in strict number order.

To use, or run, the program, you type it out exactly as it is here. At the end of each line you press the key called RETURN (or ENTER or NEWLINE on some micros). Then you type RUN and the micro will carry out the program.

1 Building a program

> You are testing a friend's go-kart. The steering fails as you are heading for a duckpond. The brake can be operated only by typing a special code letter which your friend forgot to tell you. There is time for five guesses at the letter before you get a soaking.

2

> 1. Print title and instructions
> 2. Choose random letter
> 3. Ask player for a guess at the letter
> 4. If guess is correct, print message and stop
> 5. If there is time, give hint and go back to 3
> 6. If not, print SPLASH and stop

The first stage in writing a program is to write down a detailed outline for the program in English. This outline is for a computer game.

Then break the idea down into steps and work out what the computer must do at each stage. List the steps in the correct order.

PRINT tells the micro to display everything inside the quotes on the screen.

INPUT tells it to expect a message from you and store it in a place in its memory called P$.

$ represents a "string" of characters.

CLS clears the screen.

FOR ... TO tells the micro how many times to carry out the instructions in lines 60 to 100.

IF ... THEN tells the micro what to do IF a certain condition is true. In this case, GOTO tells it to jump to line 130. If the condition is not true, the micro carries on to line 90.
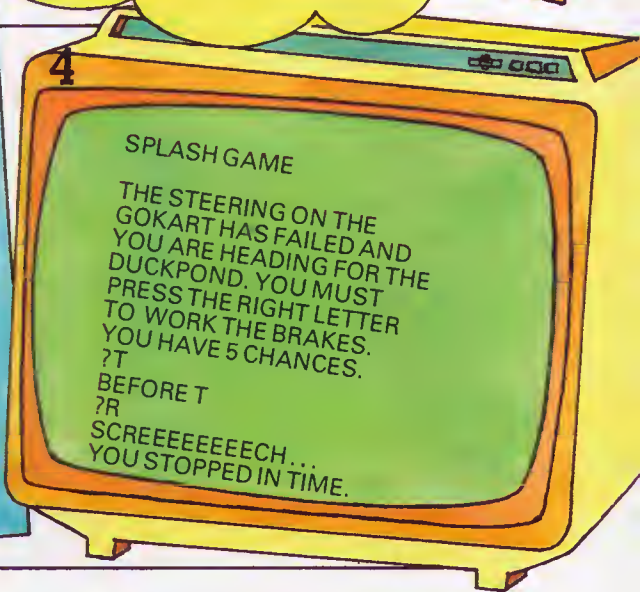
END tells the micro it has done all it has to do and the program is finished.

HALT! WHAT IS THE PASSWORD?

YOU CAN HAVE TWO TRIES.

?EGGANDCHIPS

WRONG.

?MICROCHIPS

ENTER, FRIEND

When you run the program, the micro asks you for the password and stores it in its memory. Then it asks for a guess at the password. The word INPUT in line 70 makes a question mark on the screen to show the micro is waiting for a message from you. It compares the guess with the word in its memory and if it is the same it prints ENTER, FRIEND.

3

```
10  PRINT "SPLASH GAME"
20  PRINT
30  PRINT "THE STEERING ON THE"
40  PRINT "GOKART HAS FAILED AND"
50  PRINT "YOU ARE HEADING FOR THE"
60  PRINT "DUCKPOND. YOU MUST"
70  PRINT " PRESS THE RIGHT LETTER"
80  PRINT " TO WORK THE BRAKES."
90  PRINT "YOU HAVE 5 CHANCES."
100 LET C$=CHR$(64+INT(RND(1)*26+1))
110 FOR G=1 TO 5
120 INPUT G$
130 IF G$=C$ THEN G=5: GOTO 210
140 IF G$<C$ THEN PRINT "AFTER";
150 IF G$>C$ THEN PRINT "BEFORE";
160 PRINT G$
170 NEXT G
180 PRINT "SPLAAAAAAASH"
190 PRINT "YOU HAVE GOT WET."
200 END
210 PRINT "SCREEEEEEEECH..."
220 PRINT "YOU STOPPED IN TIME."
230 END
```

These programs will not work on all micros because of the different dialects of BASIC. The most likely commands to need changing are CLS which tells the micro to clear the screen, and RND which tells it to pick a random number. If you have a micro and the program does not work, look in the manual to find how to alter it.

4

SPLASH GAME

THE STEERING ON THE
GOKART HAS FAILED AND
YOU ARE HEADING FOR THE
DUCKPOND. YOU MUST
PRESS THE RIGHT LETTER
TO WORK THE BRAKES.
YOU HAVE 5 CHANCES.
?T
BEFORE T
?R
SCREEEEEEEECH...
YOU STOPPED IN TIME.

Now translate each step of the program into BASIC. Type it into the micro line by line, checking the lines to make sure they are correct.

This is what happens when you run the program. The letters after the question marks are your guesses. There is more about running programs over the page.

13

Running programs

When you type in a listing, all the lines of the program go into the micro's memory. Instead of typing, you can "load" a program, that is, put it into the micro by using a cassette recorder. Below there are some hints on typing in programs and loading them from cassettes. If the program does not work when you type RUN, it probably has a bug (mistake) in it. Some bugs cause the program to "crash" and it stops working. Others cause unexpected things to happen in the program. You can find out about some common bugs on the opposite page.

1 Typing in a listing

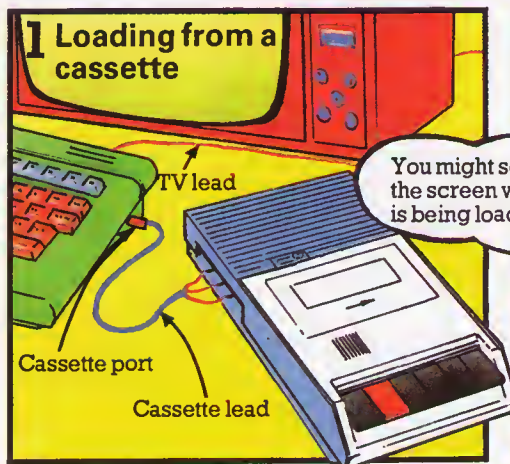Use a ruler to keep your place.
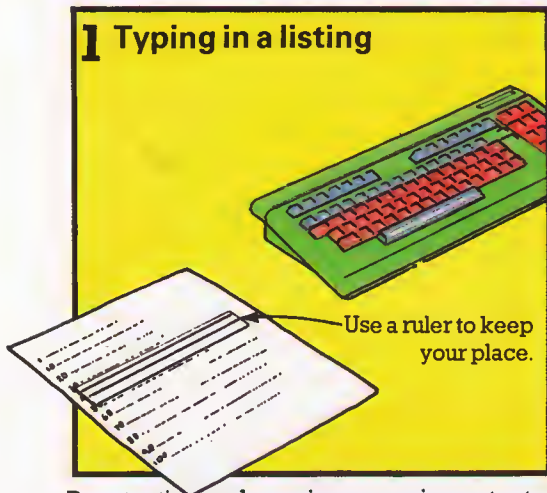
Punctuation and spacing are as important to the micro as letters and numbers, so you have to type in a listing very carefully. Each line will appear on the screen as you type for you to check.

2
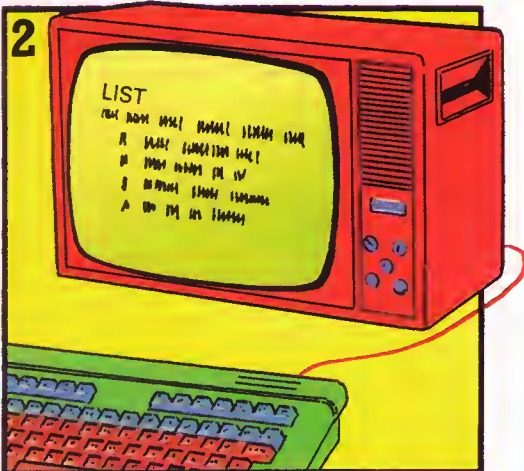
LIST

If the program does not work when you try to run it, you can type LIST to display it on the screen. You can then check it again for bugs and correct it before trying again.

1 Loading from a cassette

TV lead

You might see jagged lines on the screen while the program is being loaded.

Cassette port

Cassette lead

A program is saved on cassette tape as a series of high-pitched bleeps. To load it into the micro you connect up the cassette recorder as described in the micro's manual. You need to adjust the volume to 7 or 8 and the tone to a high treble setting so the micro can pick up the sounds.

2

LOAD "MOUSETRAP"

Try playing the cassette in the usual way to hear how the program sounds.
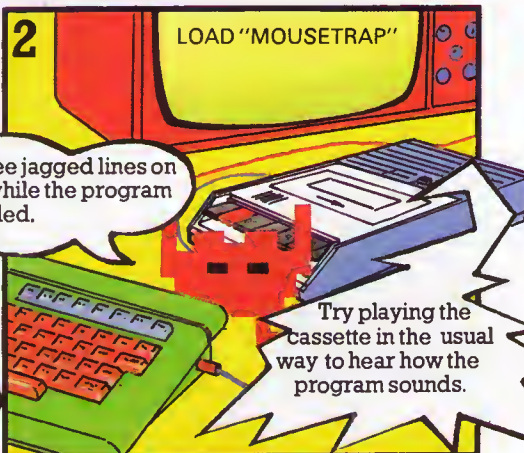
When you type LOAD and the program name in quotes, and press PLAY on the recorder, the program should be copied into the micro. This can take a few seconds or minutes, depending on the length of the program. If the program does not load successfully, the tone and volume settings might need adjusting.

14

Bugs in programs

This picture illustrates a program which has lots of bugs. The most common bugs are typing mistakes. If you do not type BASIC correctly the micro will not understand the commands. This kind of bug is called a syntax error.

Most micros send error messages to the screen when they come across something they do not understand. Some do this as you type in the program. Others wait until you RUN or LIST the program. Here are some examples of error messages.

MISSING ". The words after PRINT should be enclosed in quotes.

```
10 PRINT "HOW MANY
   CROCODILES IN THE
   RIVER?
```

```
20 "YOU HAVE FIVE
   GUESSES."
```

SYNTAX ERROR. No PRINT statement to tell the micro to put this on the screen.

```
30 LET A = 6
```

```
40 FOR N = 1 TO 5
```

CAN'T MATCH FOR. FOR . . TO . . NEXT are part of the same command telling the micro to repeat this and the next four steps five times. It is called a loop. The NEXT part of the command, which should be line 80, has no line number, so the micro does not recognize it.

```
50 ONPUT G
```

SYNTAX ERROR. The word is wrongly spelt so the micro does not understand.

NO SUCH LINE. There is no line 130 in this program.

```
60 IF G = A THEN N = 5:
   GOTO 130
```

```
70 PRINT "WRONG"
```

```
NEXT N
```

```
90 PRINT "SNAP! YOU
   HAVE BEEN EATEN UP."
```

```
100 END
```

```
110 PRINT "RIGHT. NOW
    PADDLE AWAY FAST!"
```

```
120 OK – THAT'S ALL
```

SYNTAX ERROR. This is not in BASIC so the micro does not understand. It should read END.

Here is the correct program:
```
 10 PRINT "HOW MANY CROCODILES
    IN THE RIVER?"
 20 PRINT "YOU HAVE FIVE GUESSES."
 30 LET A = 6
 40 FOR N = 1 TO 5
 50 INPUT G
 60 IF G = A THEN N = 5: GOTO 110
 70 PRINT "WRONG"
 80 NEXT N
 90 PRINT "SNAP! YOU HAVE BEEN
    EATEN UP!"
100 END
110 PRINT "RIGHT. NOW PADDLE
    AWAY FAST!"
120 END
```

15

Saving programs

After you have typed a program into the micro, you can copy it on to cassette tape. This is useful as the program in the micro's random access memory is lost when it is switched off. You can also save programs on floppy disks, using a disk drive, which is better if you want to store a lot of programs. You can make paper copies with a printer, too.

A cassette recorder, disk drive or printer plugs into a socket on the micro called a port. This contains special circuitry called an interface which converts the micro's own machine code signals into the kind of electrical signals the device uses.

Cassettes

SAVE means store the program.
LOAD means copy it back into the micro. These are BASIC words.
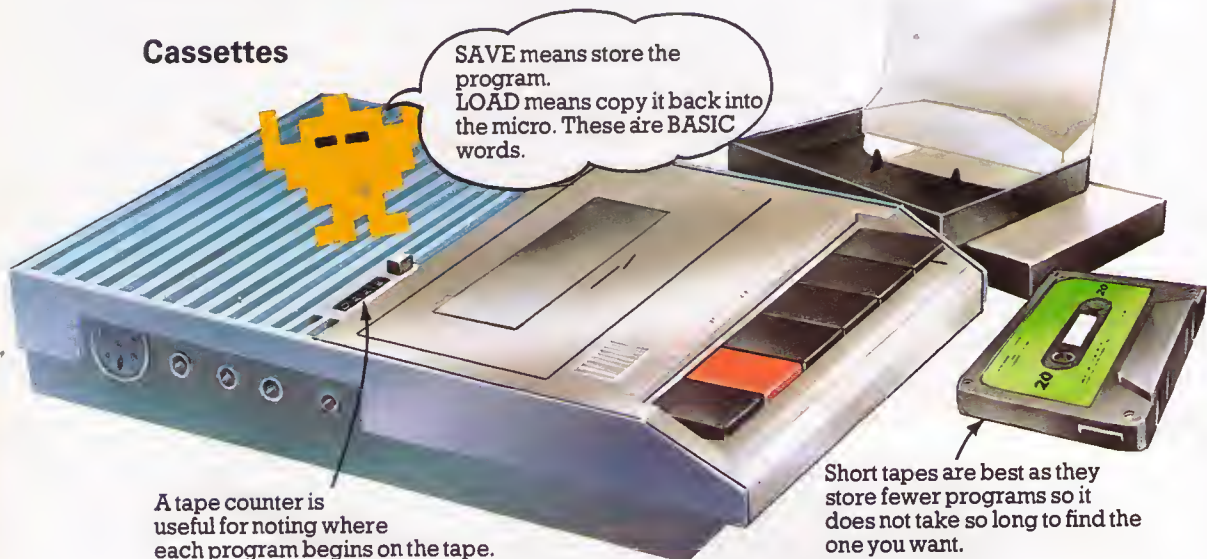
A tape counter is useful for noting where each program begins on the tape.

Short tapes are best as they store fewer programs so it does not take so long to find the one you want.

For most micros you can use an ordinary portable cassette recorder, but a few need their own special recorder. You can buy specially made "data tapes" for recording programs, but any good quality tapes will do.

Saving and loading programs on cassette can be quite tricky. If it does not work, the "recording head" of the recorder may need cleaning. If the program contains bugs, the micro will not let the recorder save it.

1 Saving programs on tape

You connect the cassette recorder to the micro as described in the manual. Make sure the leads do not cross each other or you might get interference.

2

SAVE "METEOR"
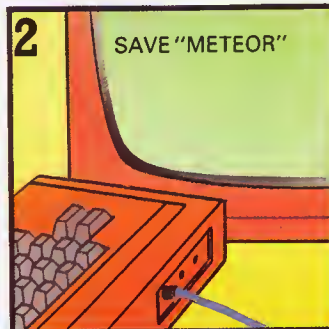
Then you type SAVE and the program name in quotes on the keyboard and press RECORD and PLAY on the recorder to save the program.
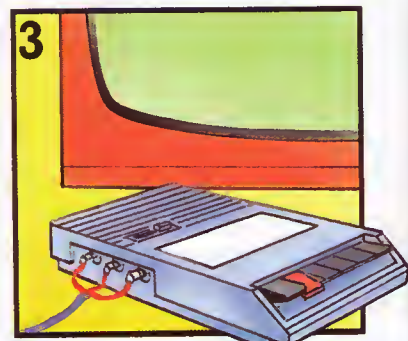
3

As the cassette tape travels past the recording heads on the cassette recorder, the program is saved as a pattern of magnetic dots on the tape.

16

Printers

This printer is very fast
and gives good quality print-outs.

This printer is
much cheaper, but it is slower
and the print-outs are not so good.

You can print out program listings, data
and sometimes even pictures with a
printer which you connect to the micro.
Most micros use a standard type of
interface called an RS232 inside the
connection.

Information saved by a printer is called
"hard copy". You can make lots of copies
of the same program to distribute among
friends. Printers can work very fast. Some
expensive ones can print out several lines
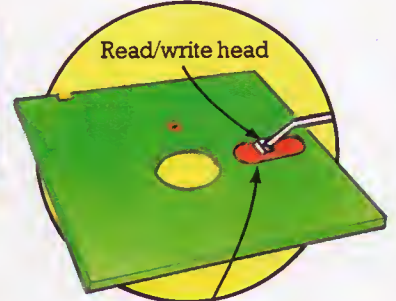per second.

Floppy disks

Read/write head
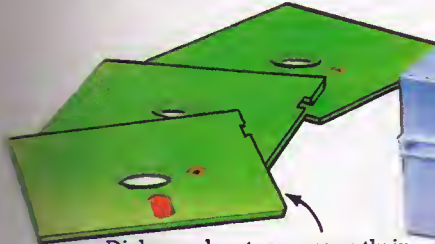
Disks are kept permanently in
protective sleeves. The sensitive
surface will be spoilt if you touch it.

Disk drive

Disk spinning round
in its sleeve.

Floppy disks store programs in the same
way as cassette tape. The disk's surface is
smooth, without grooves like a record.
Saving and loading take place inside
a disk drive which you plug into the micro.

The disk is spun inside the disk drive and
a "read/write" head moves rapidly over its
surface through a slot in the sleeve. This
head can "read" any data stored on the
disk and "write" new data on to it.

More about saving and loading

When you save a program you usually
give it a name. You need to keep a record
of program names as, when you want to
load a program and you give the micro the
name, if a letter or even a space is wrong
the micro will not recognize it.

When you load a program back into the
micro from tape or disk, a copy goes into
the micro's memory. You can then change
this copy or use different data without
altering the version stored on the tape or
disk.

17

Micro pictures

A micro makes pictures by lighting up tiny areas called pixels on the screen. Pictures made by a computer are called graphics and you can give the micro instructions for graphics by typing in a program on the keyboard. You can also make pictures by drawing on the screen with a light pen, or with a special piece of equipment called a graphics tablet. You can find out how these work below.
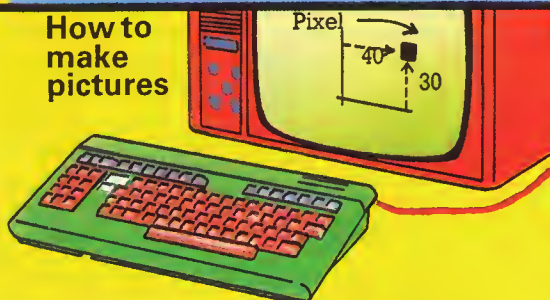
## Lighting up the screen
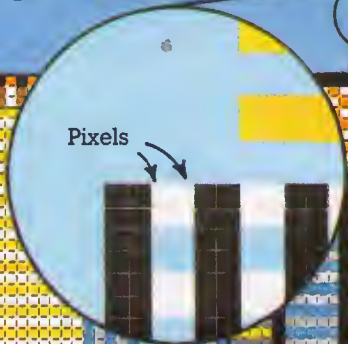
Pixels

If you look closely at a computer picture you can see all the pixels. Most computers can make colour pictures if they are connected to a colour TV or monitor, and the pictures are made by lighting up areas of pixels in different colours.

Characters (i.e. letters, numbers and symbols) are also made up of pixels. A micro divides the screen up into rows of invisible squares and each character is made by lighting up different combinations of the pixels in a square.

KING KONG

How to make pictures

Pixel

40

30

Special pen

Graphics tablet

In a graphics program you tell the micro which pixels to light up by typing in their co-ordinates. The co-ordinates for each pixel show how far it is along and up the screen, measured in numbers of pixels.
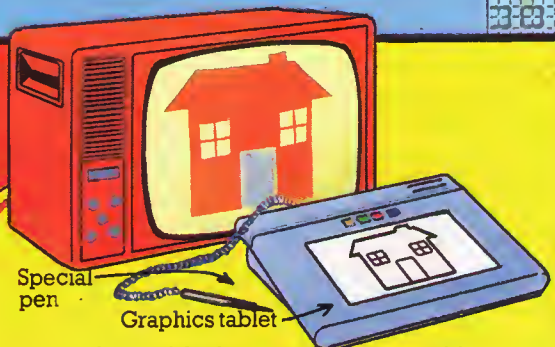
A graphics tablet has a pressure-sensitive surface covered with a grid. You place your picture on the grid, then trace over it with a special pen. This automatically gives the micro the co-ordinates for all the pixels.

1

To light up each pixel separately needs a lot of memory so most micros deal with them in groups. Each group is controlled by a separate instruction from the micro and all the pixels in one group are the same colour.

2

A micro with a large memory can make pictures with smaller groups of pixels than a micro with a smaller memory. This makes the pictures more realistic and they are called "high resolution graphics".

The number of characters you can fit on the screen depends on the number of squares and this varies from micro to micro. On a micro which divides the screen into 30 columns and 20 rows, you can fit 30 characters across the screen and 20 lines down.
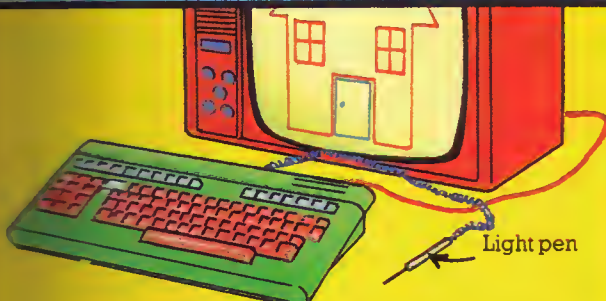
Light pen

You can draw directly on the screen by touching it with a light pen. As you draw a line, it sends messages to the micro to light up the pixels along the line. The pen can "see" the beam which lights the screen and it tells the micro the positions of the pixels in relation to the beam.

3

A micro can make moving pictures, called animations, by switching the pixels on and off, and then on again in the next position for the objects on the screen. This happens so quickly it gives the impression of movement.

Micro sound

Most micros can play tunes and make sound effects, and some can even speak words. Micros which can make sounds usually have a special chip called a synthesizer inside the keyboard. For some micros you can buy a synthesizer unit separately.

You can tell the micro which sounds to make by typing in a command such as SOUND or BEEP, followed by numbers indicating the note you want (e.g. C or B) and how long you want it to be played. You can find out how the micro makes sounds in this picture.

1 Making music

You can buy music programs on cassette – some programs make screen pictures at the same time.

You can program a micro to play a tune by giving it instructions for each note. Some micros can also play chords and harmonies. These have several "voices" which can each be programmed to play different notes at the same time.

2

Another way to tell the micro which notes to play is with a light pen. You give the micro a program which makes a stave (the lines for musical notes) on the screen, then draw the notes you want with a light pen.

Your instructions to the micro.
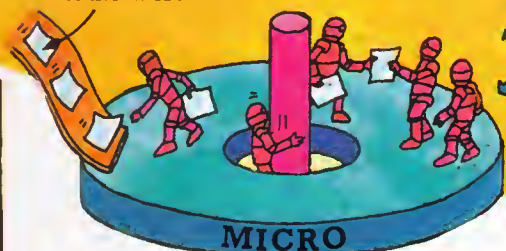
MICRO

Micro's message to the synthesizer.

Electric signal from the synthesizer.

SYNTHESIZER

When you type in an instruction for a sound the micro sends a message to the synthesizer in machine code, telling it which sound to make. The synthesizer produces an electric signal which is strengthened in an amplifier and then sent

More about sound

Sound wave

Wave for loud high note
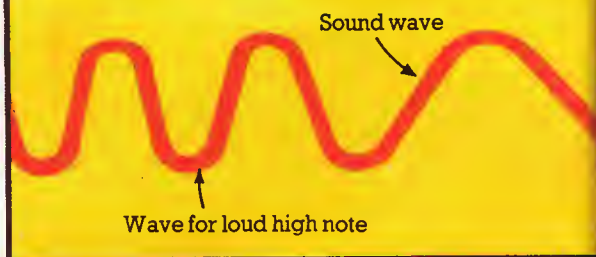
The vibrations in the air made by a loudspeaker are called sound waves, and different sounds have different shaped waves. For instance, a loud, high note has tall, squashed-up waves. The height of the waves shows how loud the note is. The

You can program some micros to make sound effects like marching feet or a telephone ringing.

HELLO

BOOM!

LOUDSPEAKER

Some micros have a loudspeaker in the keyboard. Others use the TV loudspeaker and you can control the volume with the TV volume control.

AMPLIFIER

on to a loudspeaker. The signal makes the loudspeaker vibrate and this makes the sound. Different signals from the synthesizer make the loudspeaker vibrate at different rates and this makes different sounds.

Wavelength of wave

Height of wave

Wave for loud low note
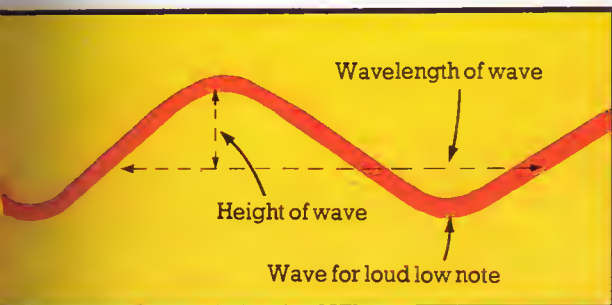
wavelength of the waves, that is, how close together they are, shows how high or low the note is (this is called the pitch). The variations in the volume and pitch of a sound over a period of time is called the sound envelope.
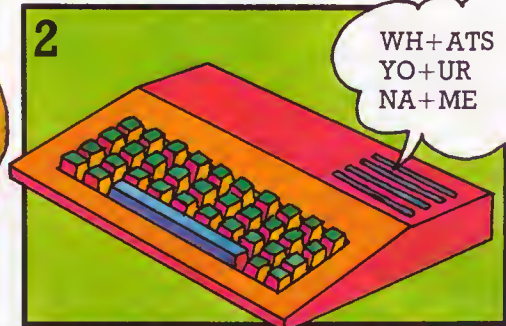
1 Talking micros

TH+AT
F+AT
C+AT
EA+TS
RO+BO+TS

It is more difficult for a micro to speak than make music as the sounds in words are more complicated. Most words are made up of several sounds, e.g. RO-BO-TS. Micros which can talk have the word sounds stored in machine code in a special chip.

2

WH+ATS
YO+UR
NA+ME

Using a synthesizer, the micro puts the word sounds together to make words, according to grammar rules stored in its memory. This is called speech synthesis. Micros with speech synthesizers are useful for blind people who cannot see a screen, or children who cannot read.

3

? ...

This town ain't big enough for both of us.

It is much more difficult for computers to understand speech. They have to be programmed to recognize all the word sounds. As people have different voices and pronounce words differently, only a computer with a huge memory can store enough information.

21

Inside the keyboard

The picture on these two pages shows the parts inside a small computer. All computers have the same basic parts as those shown here, although most are more complicated and have more components.

   The most important parts in the computer are the chips – the four black boxes on legs. All the work inside the computer is done by electrical signals pulsing through the chips and flowing along the metal tracks on the printed circuit board. You can find out more about how the computer works on the next few pages.

ROM chip
The permanent program of instructions telling the computer how to operate is stored in here.

Voltage regulator
This converts the 9 volts from the power supply into the smooth, regular 5 volts which the micro uses.

Printed circuit board (PCB)
This has metal tracks laid out on its surface and the electrical signals in the computer flow along the tracks between the chips. There are other electronic components on the board called capacitors and resistors and these help control the flow of electricity.
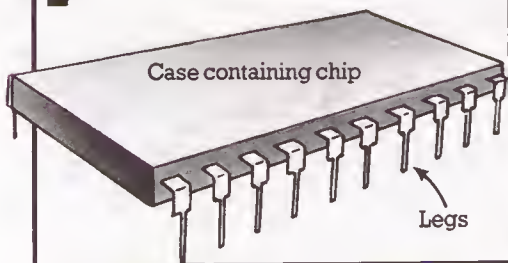
Resistors

RAM chip
This is the random access memory, where the programs and data you put into the computer are stored.

1 Looking at a chip
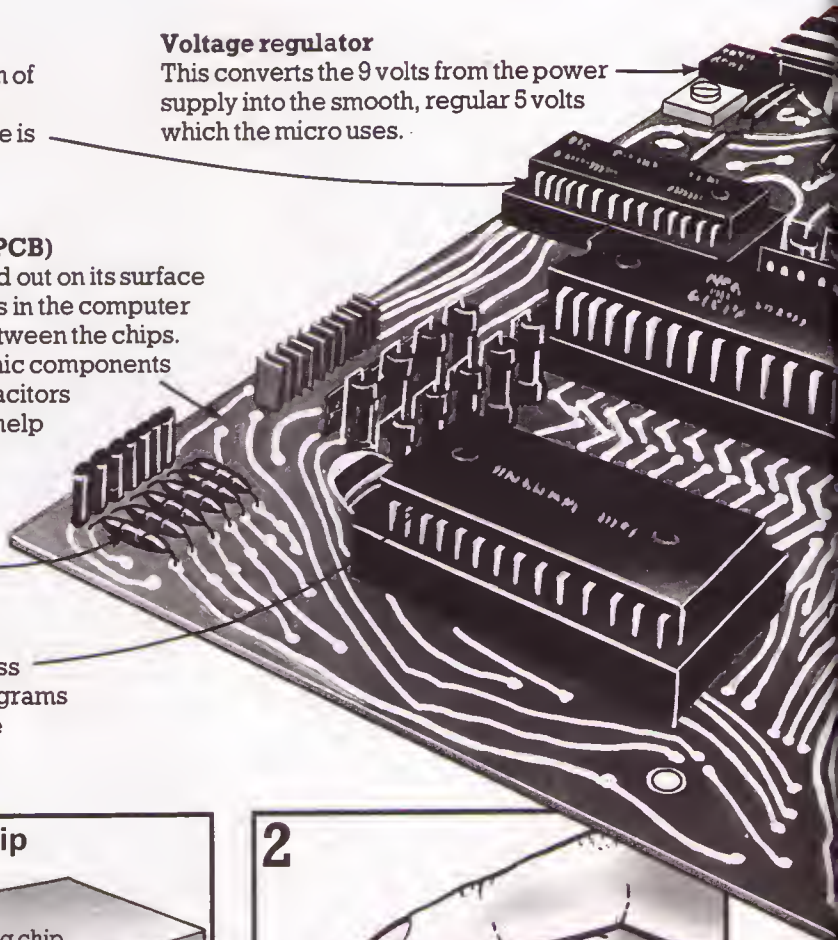
Case containing chip

Legs

A chip is a small box containing a tiny chip of silicon. The surface of the chip is covered with further circuits which are minute and very complicated. The metal legs on the chip's case carry electrical signals to and from the chip.

2

Chip of silicon
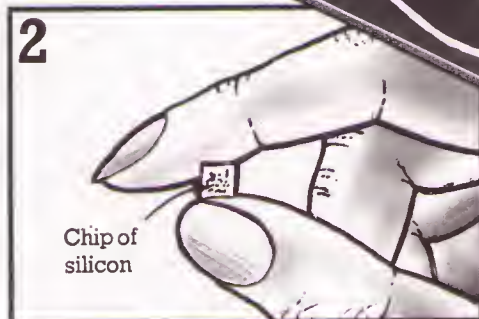
This picture shows the actual size of a chip. It is about as thick as a fingernail and may have as many as ten different circuits engraved in it. The proper name for a chip is "integrated circuit" or IC.

Sinclair Computer Logic chip
This is a special chip which contains extra operating instructions for this computer.

Sockets for connecting the TV and power supply and for other equipment such as cassette recorder or printer.

Modulator
This converts the computer's signals into signals the TV can understand.

Capacitor

Microprocessor
This is the central processing unit (CPU), the control centre of the computer. It carries out the instructions in your program and controls the flow of information to the RAM and TV screen. It contains a quartz crystal "clock" which pulses over a million times a second and regulates the flow of electrical signals inside the computer.

Edge connector
This is where you plug in equipment such as an add-on memory pack or program cartridge. Metal strips at the edge of the board carry electric signals to and from the memory or cartridge.

The circles show where the metal tracks pass through the printed circuit board and continue on the other side.

More powerful computers

1

CPU

PCB

ROM chips

RAM chips

More powerful computers have larger memories and more chips. The picture shows the PCB of another micro with about 40 chips on it. There are several ROM and RAM chips and this gives the micro a larger memory.

2

Cabinets

A really powerful computer, such as those used by large companies, has hundreds of PCBs covered with chips. The PCBs are stored in cabinets and all the cabinets may fill a room. This is called a mainframe computer and it can carry out many tasks at the same time.
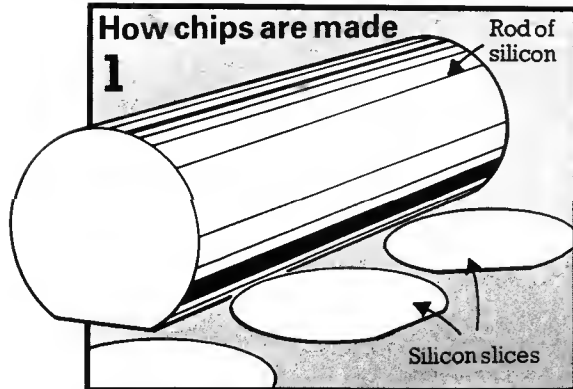
3

A minicomputer is a smaller version of a mainframe. It has several cabinets with PCBs and it is usually specially designed to do one particular kind of work, such as accounting or storing the information for a databank.

23

Inside a chip

Each of the chips in a computer has circuits specially designed for the particular jobs it has to do. The picture on the right shows two silicon chips, much enlarged. One is a microprocessor and the other is a ROM chip and you can see the patterns of the different circuits on each chip. The circuits are so small and intricate that when they are tested during manufacture up to half of them have to be discarded because they are faulty.
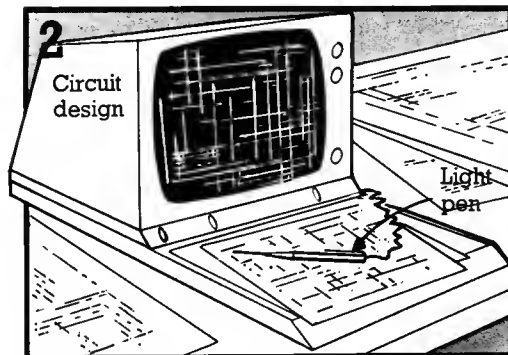
Microprocessor ▶

A microprocessor is sometimes called a computer on a chip. It has several different kinds of circuits and in fact can do the work of a tiny computer. The microcomputer is named after the microprocessor.

These are RAM circuits for the microprocessor's temporary memory. Information the microprocessor needs for a particular job is stored here.

RAM

This is a ROM circuit. It contains instructions telling the microprocessor how to operate.

ROM

How chips are made

1

Rod of silicon

Silicon slices

Chips are made from very pure silicon crystal. The crystal is shaped into rods and then cut into slices about 100mm in diameter and 0.5mm thick. Each slice will make about 500 chips. Silicon is made by purifying sand, and so the chips are quite cheap.

The circuits on the microprocessor are connected by tracks called "busses". Tracks which continue on to the printed circuit board to connect the microprocessor to other chips are also called busses.
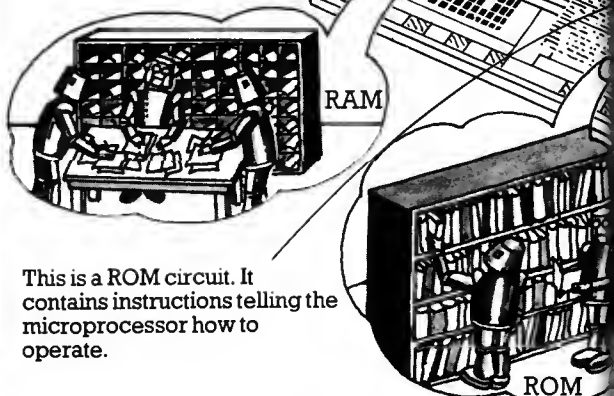
2

Circuit design

Light pen

Nowadays computers help to design circuits for chips. Here, a light pen is being used to make alterations to a circuit design. Next it will be reduced in size to fit on a chip.

3
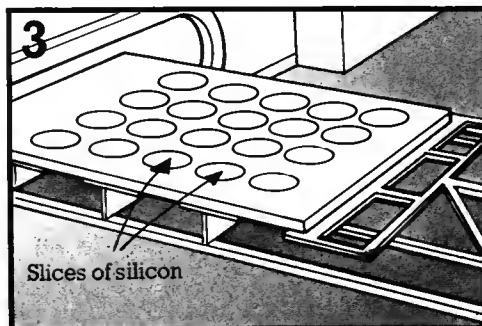
Slices of silicon

The circuit designs are placed on the chips by a photographic process and the slices of silicon are put in a furnace. There, the circuits are chemically etched into the silicon.
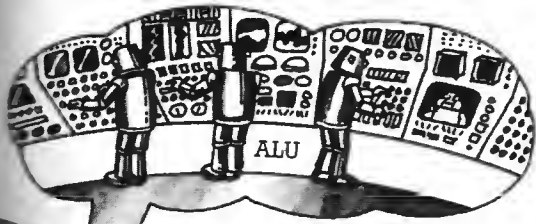
All the computer's calculating and processing is done in the arithmetic and logic unit circuits (ALU).

:0058

The clock controls the rate at which pulses travel round the microprocessor.

▼Memory chip

The circuits on a memory chip are like hundreds of little boxes. On a ROM chip, each box contains a piece of information, but on a RAM chip the boxes are empty until you put the information in.
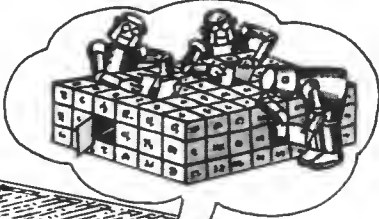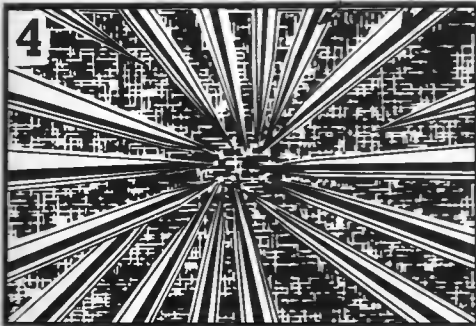
Other uses of microprocessors
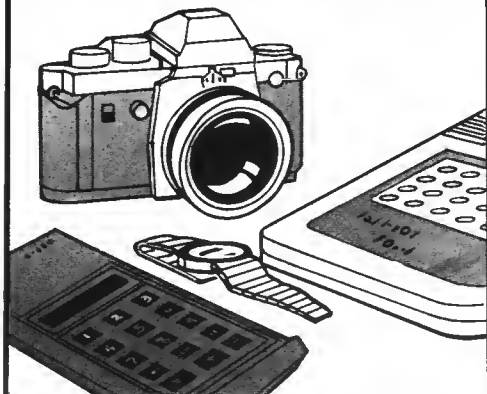
Microprocessors are used as control mechanisms in all sorts of equipment. They are tiny and very light, and so can be put in things like cameras, watches and pocket calculators.

Microprocessors have replaced old-fashioned, bulky electronic devices in many everyday things such as washing machines and telephone switchboards. They are more efficient and reliable.

Probes

4

Many different circuits can be etched into the same chip, and the process can take several weeks. The finished chips are tested on the slice with tiny probes under a microscope, and faulty ones marked.

5

The silicon slices are then cut up into individual chips with a diamond saw and the faulty ones thrown away. The perfect chips are then packaged into protective cases which can be fastened on to a PCB.

25

How chips work

The circuits in a chip contain thousands of tiny components called transistors through which the current flows in rapid pulses. Some of the tr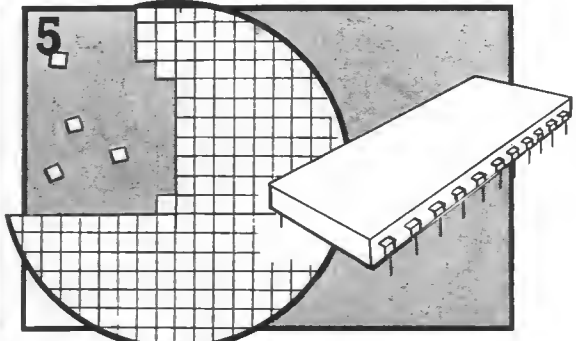ansistors are combined to form "gates". Some gates allow pulses through and some do not. This creates patterns of pulse signals and "no-pulse" signals, which make up machine code. The no-pulse is just as important as the pulse signal.
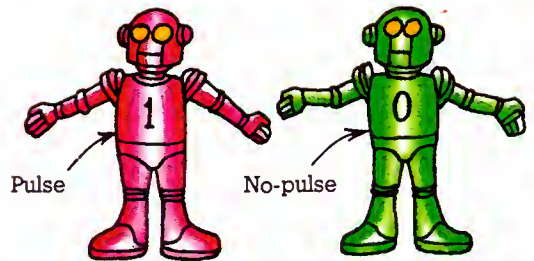
Pulse   No-pulse

Machine code is made up of only two signals – pulse and no-pulse. Codes made up of two signals are called binary codes. The signals are represented by 1 and 0. Below you can see how binary works.

Counting in binary

$(1 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1) = 13$
13 is written as 1101 in binary.

Binary numbers are made up from two digits, 0 and 1. They are written in columns of ones, twos, fours, eights, and so on. You make up the numbers by putting 1s and 0s in the correct columns.

$(4 \times 1000) + (0 \times 100) + (2 \times 10) + (1 \times 1) = 4021$
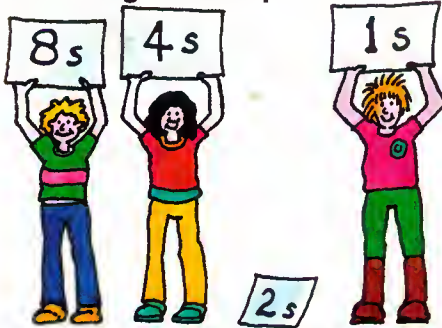
The decimal counting system we use works on the same principle as binary, but we use ten digits (0 to 9), probably because we have ten fingers. Decimal numbers are written in columns of ones, tens, hundreds, and so on.

More about machine code

One bit

Eight bits = one byte

Streams of bytes represent all the information a computer uses.
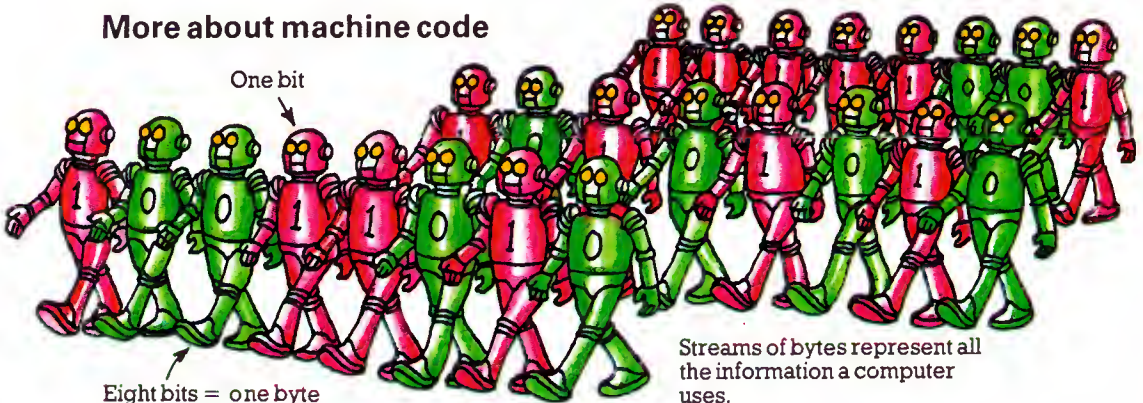
Each pulse or no-pulse signal is called a "bit", short for binary digit. Most micros use groups of eight bits to represent pieces of information. A group of eight bits is called a "byte" and is rather like a word made up of eight letters.

There are 256 different ways of arranging the 0s and 1s in an eight-bit byte. This is enough to represent each symbol on the keyboard by a single byte, with some left over for things like colours and sounds.

How the computer processes information

The computer processes information by sending the pulse signals which make up machine code bytes through different combinations of transistors, called gates. These alter the patterns of the pulses they receive in a particular way. The points where they receive signals are called terminals. Some gates receive two signals but only send on one. Here are three kinds of gate:

An AND gate sends a pulse on if it receives one at both its terminals.

An OR gate sends a pulse on if it receives one at both or either of its terminals.

A NOT gate has one terminal. It only sends a pulse on if it does not receive one.

How the computer adds up

These pictures show how the computer uses a particular arrangement of gates to add up any binary digits (1+1, 1+0, 0+1, 0+0). The computer does all its processing using sets of gates like this, though this is a very simple example.

This is how the computer adds 1 and 1.

AND gate receives two pulses and sends one on.

Each pulse signal sent to both gates.

OR gate receives two pulses and sends one on.

NOT gate receives one pulse and does not send one on.

Pulse signal sent to NOT gate and down output wire.

AND gate receives one pulse and does not send one on.

Wires along which the pulses travel forward.

10 in binary is the same as 2 in decimal.

This is what happens when other binary digits are added together in the same set of gates.

27

More about chips

The way a micro works depends on the kind of chips it has inside it. Micros with the same microprocessor understand the same version of machine code. The interpreter which translates BASIC into machine code is stored in the ROM. Micros with the same ROM chip usually understand the same dialect of BASIC. This is known as software compatibility.

BASIC is more like human language than machine code, so the micro needs a large interpreter. Languages like BASIC are called high-level languages. Low-level languages are more like machine code, and are easier for the computer to translate.

Microprocessor chips

Z80 chip

6502 chip

There are lots of different kinds of microprocessors, but the two most commonly found in home micros are the Z80 and 6502 chips shown here. Th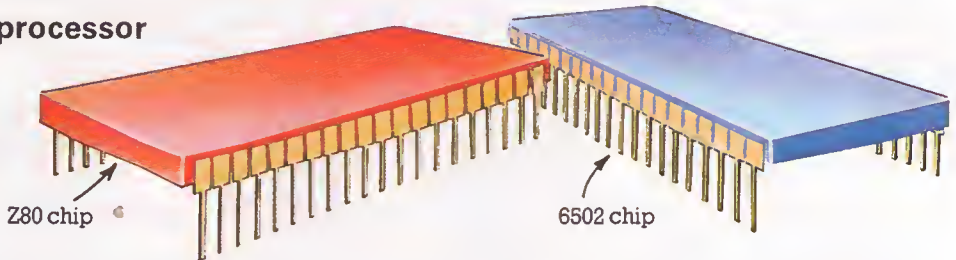e differences between them lie in their circuitry, so it might be difficult to tell them apart just by looking at their outside cases. The micro's operating instructions stored in the ROM chip have to be written in the correct version of machine code, for instance, in Z80 machine code for the Z80 microprocessor.

Inside the ROM

The ROM consists of tiny areas with number addresses which each store one byte of information. You can ask a micro to show you the bytes stored in some areas by typing PEEK and an address. The manual will tell you which addresses you can PEEK into. The byte will appear on the screen as a decimal number.

MONITOR
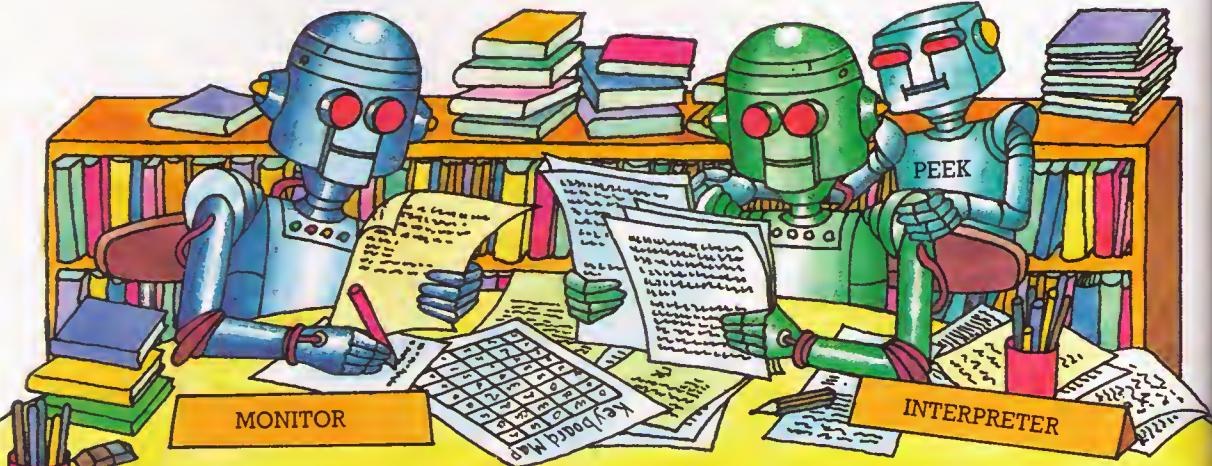
INTERPRETER

The special program which handles the running of a micro, called the monitor, is stored in the ROM along with the interpreter. One of the monitor's tasks is to detect which key on the keyboard has been pressed. It receives an electrical signal from the key and refers to a keyboard map to find which byte represents that key. Most micros conform to ASCII (American Standard Code for Information Interchange) for which bytes represent which symbols.

Inside the RAM

System variables	This area contains information for the micro, such as where the next character will appear on the screen.
Program area	This is where the program is stored.
Display file	The micro puts a machine code copy of what is displayed on the screen here.
Variables	This is where data is stored.
Current line and work space	This contains the line being typed.
Calculator stack	The CPU does some of its sums here.
Spare area	This is free space for you to use, or for other RAM areas to "borrow" if they get full.
Machine stack	The CPU uses this area for storing things, such as program line numbers.
GOSUB stack	The program line number the micro must return to after being sent on a diversion is stored here.

The RAM is divided into areas storing different kinds of information. You can PEEK into RAM in the same way as you can into ROM. You can also change the bytes stored in some areas of RAM by typing POKE followed by an address and a number. (You cannot do this with ROM as it is a permanent memory.) The micro's manual will tell you which areas of RAM you can POKE without interfering with its other jobs. You can usually POKE into the "system variables" area, and you can store things in the spare area, and retrieve them with PEEK.

Low-level programming

If you program a micro in machine code, it can act on the instructions immediately, without first having to translate them. This is useful in fast games programs, for example. Programming in streams of binary digits is complicated, though, so you can use other low-level codes, such as hex or mnemonics, instead. These are like fast, shorthand versions of machine code.

One byte

Decimal value of group is 13 – called D in hex.

Hex equivalent to the byte is D5.

Mnemonic codes

MPY = multiply

STO = store

ADD = add
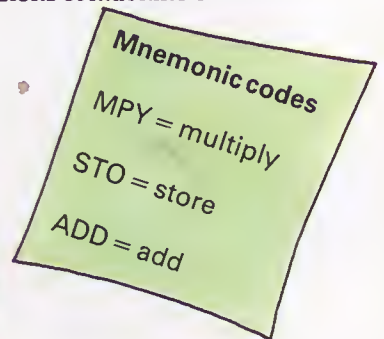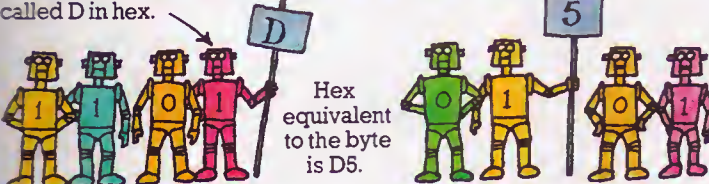
Hex, short for hexadecimal, is a number system based on 16 digits – 0 to 9, and A to F which represent the numbers 10 to 15. An eight-bit byte can be written as two hex digits. You divide the byte into two groups of four binary digits, and turn each group into a single hex digit.

A mnemonic code is a set of abbreviations which stand for certain instructions to the micro. Each mnemonic sets off a particular chain of activity in the micro. Low-level codes are easier for the computer to convert into machine code, so the interpreter can be smaller.

The story of the micro

The first true electronic computers were built in Britain in World War II. Unlike earlier mechanical adding machines, they were programmable and had memories. These computers were used by scientists to crack enemy codes and plot the flight paths of shells. Information about them was kept top secret for many years.

When peace came, a few big business corporations and governments began to use computers, but no one else could afford them. Since then, computers have got smaller, cheaper and more powerful. This has led to the development of the micro, which is a computer anyone can use – not just scientists.
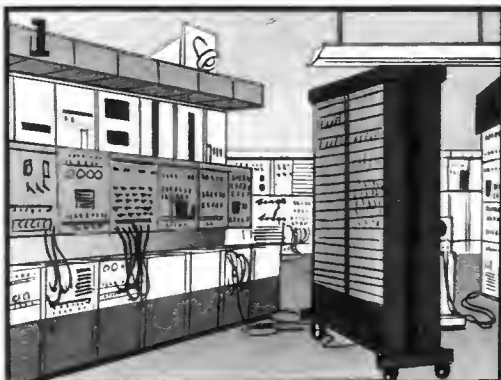
The first computers were built before transistor switches were invented. They used valves instead. These were about seven centimetres high and made of glass, and there were about 18,000 in a computer. They frequently failed, and teams of engineers were needed to locate dud valves in the complicated circuits.
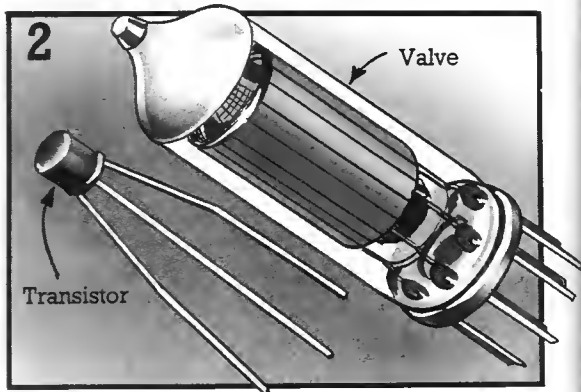

In the 1950s transistors were invented in the USA. They did the same job as valves, but were smaller, cheaper and faster. Valve manufacturers lost their scientists to new transistor companies. Soon transistors were replacing valves in all kinds of electronic equipment, such as radios, as well as computers.

By the 1960s, the US government was competing in the space race and needed small, powerful computers for their spacecraft. They financed research into "integrated circuits", which were a new invention and consisted of several transistors combined in a tiny sliver of silicon, nicknamed a chip.
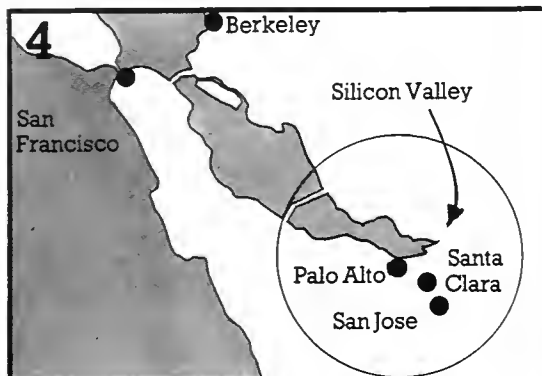
Silicon chips were an enormous breakthrough, and led to a new science called microelectronics. The main centre of research was the Santa Clara Valley in California, which became known as Silicon Valley. Microelectronics engineers learnt how to pack more and more components on to the same chip.

Computer generations

ENIAC took up the space of a house and weighed 30 tonnes.

It cost over half a million dollars to build.

It consumed 200 kilowatts of electricity.

It could add two numbers in three millionths of a second.

A valve would fail every seven or eight minutes.

1945

The history of computers can be divided into four generations, each smaller and more powerful than the last. Huge valve computers were the first generation. One of these was called ENIAC. It was completed in 1945 after taking two years to build. The second generation used transistors. Computers with chips were the third generation and the invention of microprocessors and further miniaturisation brought the fourth.

A chip is smaller and thinner than a contact lens.

It costs under five dollars.

It can add two numbers in one ten-millionth of a second.

It almost never breaks down.

It consumes only a tiny amount of electricity.

1980

5 Microprocessor chip
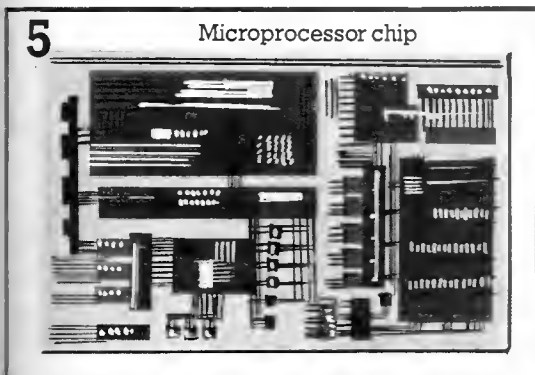
A major breakthrough came in 1971 when it became possible to place all the main electronic parts of a computer on to one chip. This was called a microprocessor. A computer circuit that would once have filled a whole room with thousands of valves could now be contained in a 5mm square silicon chip.

6

This new technology led to the production of microcomputers, which a small business or even a single person could afford. Micros were on the market by the late 1970s. Now you can buy a micro the size of a book which costs as little as a few of the valves contained in one of the earliest computers.

31

Computer chains

You can link a micro to another computer anywhere in the world, provided they have the necessary connections and there is a way of transmitting the signals between them clearly. They can use existing means of communication, such as telephone lines and satellites. The computers usually need special programs to help them understand each other, as they might use different languages or dialects, or work at different speeds. People link computers together to share information or programs. Anything in one computer's memory can be copied into another.

Using networks
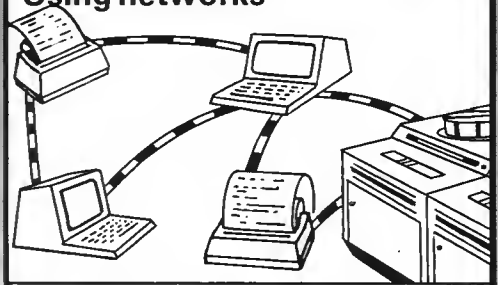
Computers can be linked together in "nets", or networks, usually using a telephone. You can find out how to do this on this page. You need a password to tell the other computer to receive your messages. You can link up with any computer equipment, for example, lots of micros could share the same printer.

Dialling a computer

1
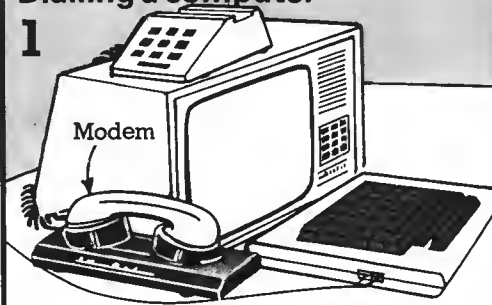
Modem

Micros can be linked using a telephone and a device called a modem*. This converts machine code signals into the kind of electronic signals that telephone lines can carry.

2
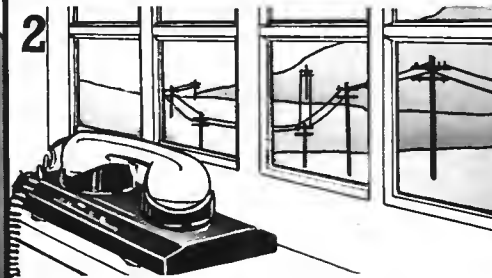
The person receiving the messages needs another modem attached to their micro to convert the signals back into machine code again.

Telephone line

Fibre optic cable
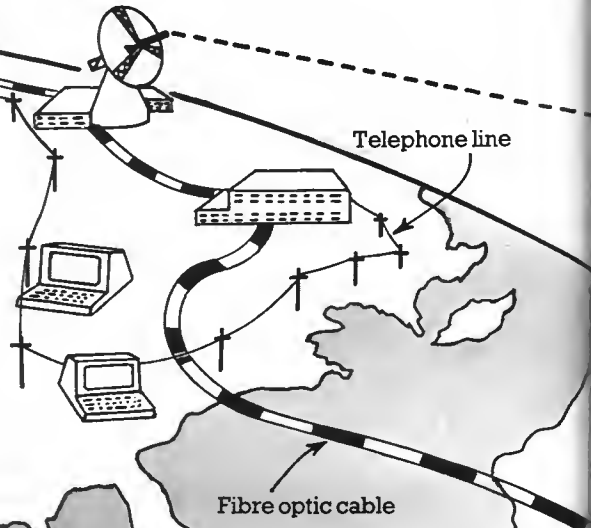
Sending signals around the world

Computer signals can be sent by satellite in the form of radio waves which bounce off the satellite and land at a particular spot on earth. This kind of satellite also transmits telephone calls and TV programmes round the world.

New ways of sending electrical signals at the speed of light are being developed using fibre optics. Machine code signals are converted to flashes of light and fibre optic cables carry them overland or under the sea to anywhere in the world.

*Modem stands for modulator/demodulator.

SUPERMARKET SEAFOOD

FRESH TODAY

1. SQUID   1.10 PER KG
2. PRAWNS   0.60 PER KG
3. MUSSELS   0.70 PER KG

DEAR JOHN,
    PERHAPS YOU WOULD
LIKE TO EXPLAIN YOUR
BEHAVIOUR WITH THAT
BOWL OF SOUP

You could do your shopping by computer, and in the future you probably will. You link a micro to a shop's computer and a display of goods for sale appears on the screen. You type in your order and give your bank account number. The shop's computer arranges delivery and contacts your bank to charge your account.

Micros linked by telephone can be used as an electronic mail system. Instead of writing a letter and posting it you can type it out on your micro's keyboard, dial a connection with someone else's micro and leave the letter on the screen. This is much quicker than using ordinary mail.

Satellite

Radar screen to collect signal.
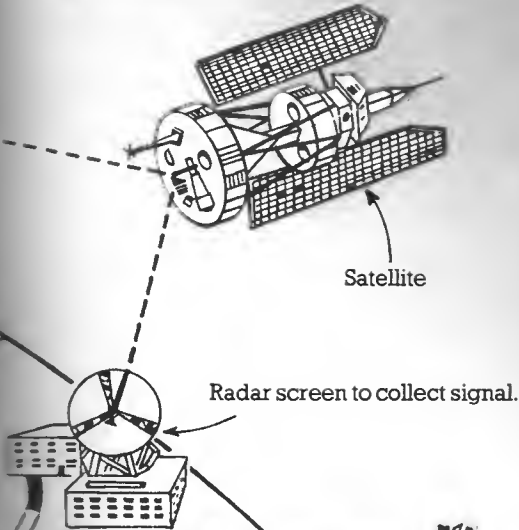
Business people can work at home using a micro in a network to communicate with a central computer in their office. They will have access to files and be able to send messages to colleagues in the network.

Nowadays, more and more micros are being connected to computerized information centres called teletext systems. With a worldwide network of computers storing and exchanging information, you can have almost any knowledge at your fingertips.

Already in some schools, micros on the pupils' desks are connected up to a central one. The teacher uses this to keep in touch with what each pupil is doing and to supply programs. The pupils can work at their own speed.

Micro control

Most micros can control other electrical equipment as easily as they control their own screen or printer, provided they have the right "ports", or sockets where you plug the e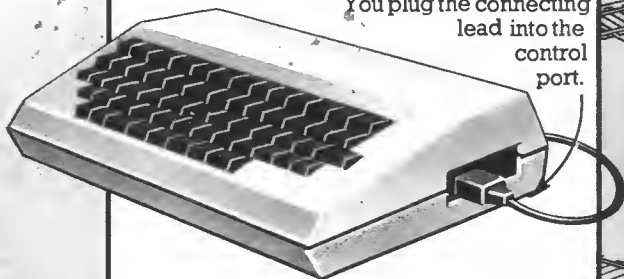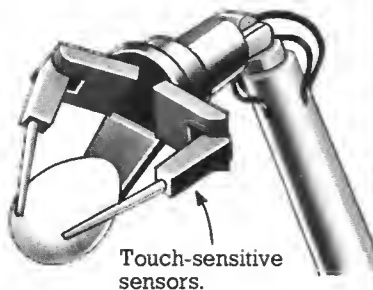quipment in. The micro's machine code signals must be converted into a form the equipment can use. This conversion usually takes place at the control port on the micro's keyboard. the part which does the controlling is the microprocessor.

Running a model railway

Here is a railway circuit controlled by a micro, which is connected to the track by a lead from the control port. It sends signals to the track to change the points and stop and start the train.

Getting signals in and out

You plug the connecting lead into the control port.

The micro needs a way of sending signals out to whatever it is controlling, and of getting progress reports back. The control port contains the interface* which handles this information. If your micro does not have a control port, you can usually buy one for it.
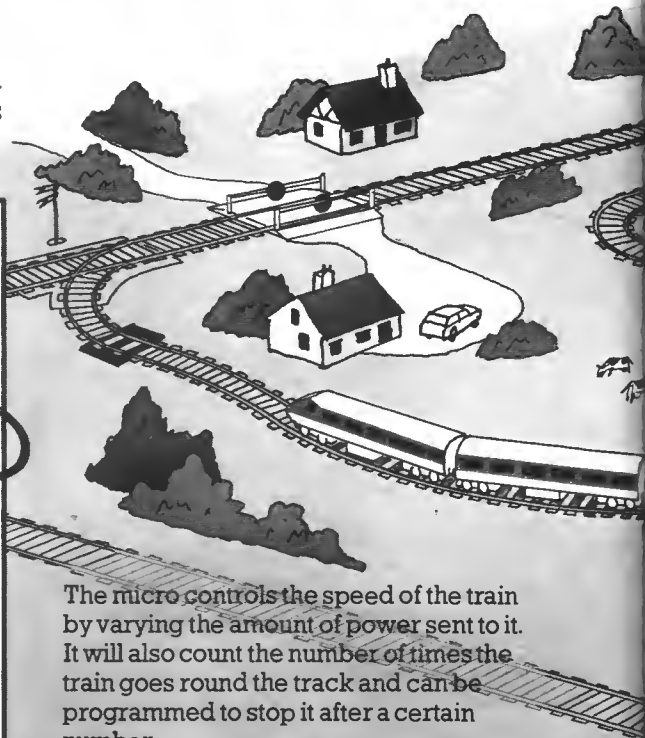
The micro controls the speed of the train by varying the amount of power sent to it. It will also count the number of times the train goes round the track and can be programmed to stop it after a certain number.

Touch-sensitive sensors.

The micro can use sensors to tell it what is happening, for instance, if it needs to know the position of something it has to move. A robot arm controlled by a micro might have touch-sensitive areas to tell it when it contacts something, or it might have a light-sensitive "eye".
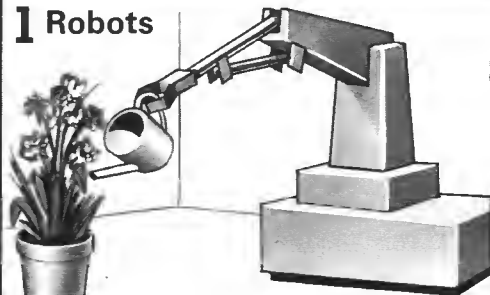
1 Robots

You can turn some micros into robots by connecting a special metal "arm", and programming the micro to make it move and pick things up. The microprocessor in the micro acts as the "brain" of the robot, using messages from sensors in the arm to help work out the next move.

34

*The connection which passes information between the micro and what it controls.

Micro controlling the railway.

Control port

Sensor

Points
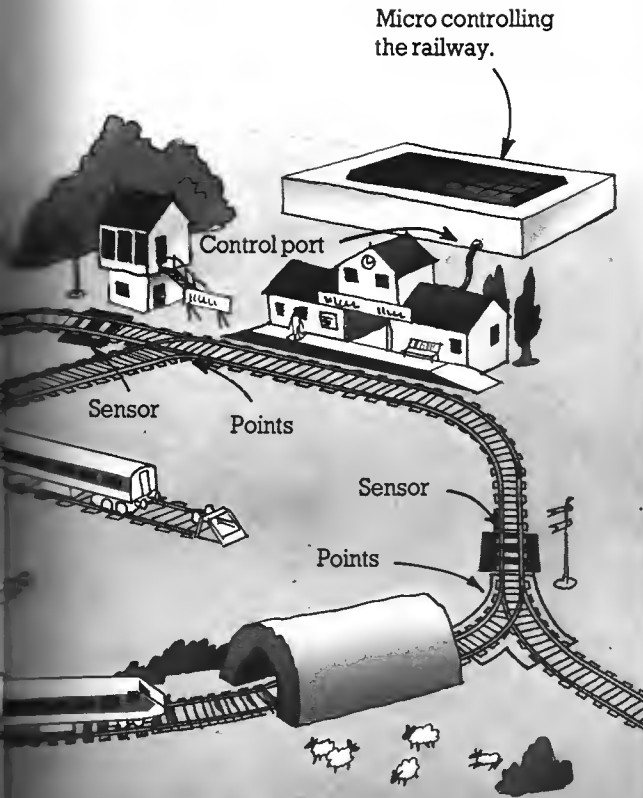
Sensor

Points

When the train crosses a pressure-sensitive sensor, a message is sent to the micro telling it which part of the track the train is on and which set of points it is approaching. The micro has a program telling it what to do next.

Micros in space

The Space Shuttle carries a microcomputer similar to an ordinary home micro. Unmanned interplanetary probes like the Viking and Voyager missions to Mars and Saturn were controlled by micros on board linked by radio to large computers on earth.

These micros carry out complicated calculations very quickly. They plot courses and control engine thrust and fuel consumption. They can monitor experiments and supervise photography. They radio reports to Earth and receive instructions back.

Larger robots are used in factories. They do lots of jobs, from moving heavy car bodies about to putting together tiny mechanical parts. These are robots and not just machines because they can be programmed to do different things and can make some of their own decisions.

3
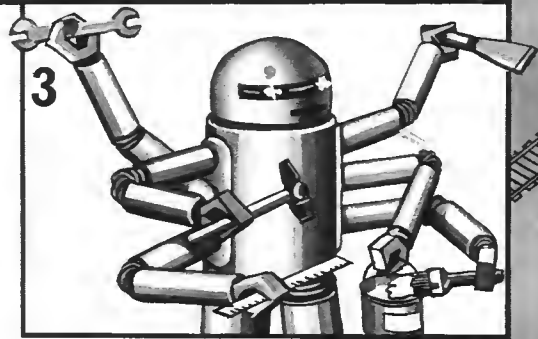
"Robot" is the Czech word for worker. It was first used to describe artificial men by the Czech playwright Karel Capek in the 1920s. Robots can be used for boring or dangerous jobs. They do not breathe, so they can work in space or in mines where there are poisonous gases.

Other micro users

Micros are used for all kinds of jobs. they are small and powerful and can work on any information once it has been converted into machine code. They process information and calculate much faster than a human. They can store lots of information in a small space, and have totally accurate memories, unlike humans.

Micros are often used for analysing data. They store sets of information and compare it to input data.

Micros in medicine


```
DOES YOUR HEAD HURT?
YES
IS YOUR VISION AFFECTED?
YES
HAVE YOU EVER HAD
MIGRAINE?
NO
```

As well as keeping medical records, some doctors use micros to help with their diagnoses. A patient types in answers to questions and the micro compares them to lists in its memory. It gives possible diagnoses and cures.
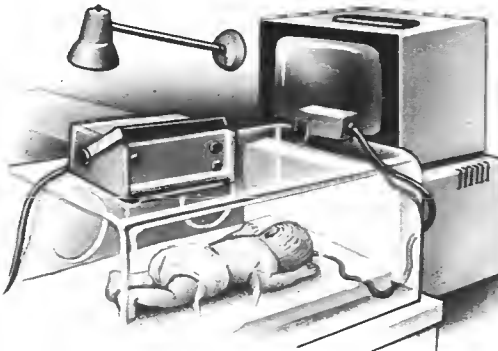
Staff at the Hammersmith Hospital in London developed a micro-based system to care for premature babies who have difficulty breathing and need their lungs artificially inflated. Too much air forced in can damage the lungs. Too little can cause brain damage. The micro monitors a baby's lungs so it gets just enough oxygen.

Weather forecasting
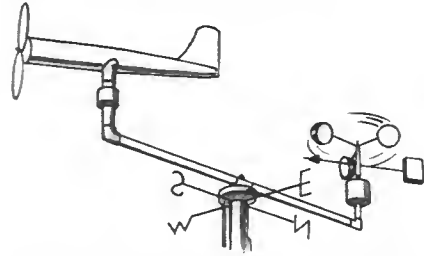
Microcomputers at local weather stations process data received from instruments, and send the results to a central meteorological office.

Helping the disabled

People who cannot speak or hear can use micros to communicate. There are special keyboards for semi-paralysed people which require only a slight movement of a finger or some other part of the body to select a word or letter.

Computer-aided design
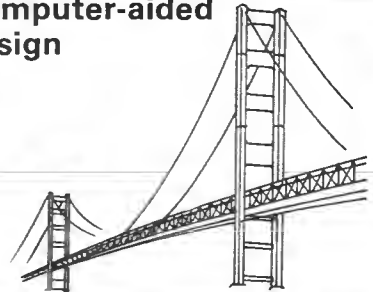
A micro can show objects in 3-D and rotate them so the designer can look at them from another angle. An architect designing a bridge or a building can ask the micro to calculate stresses and decide if it would be safe.

Portable micros

Hand-held micro

This micro can be carried like a briefcase.
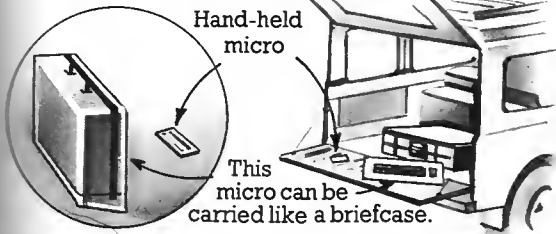
People doing field work, like a geologist prospecting for oil, or a building site foreman, might use a portable micro. It can store and process facts on the spot.

Brewing beer

Barley malt
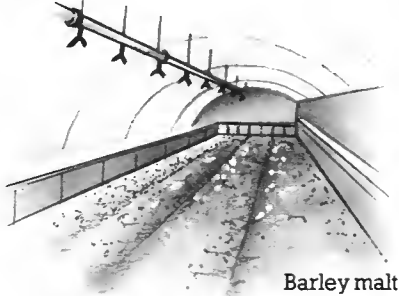
Micros are used in automated breweries and other factories. Making beer involves mixing and fermenting at precise temperatures for set times. Sensors tell the micros when one stage is complete and another ready to begin.

Learning with micros

Micros are used to teach anything from French to navigation. You can even "dissect" a rabbit on the screen using a light pen instead of having to cut up a real animal.

Micros in business

FRONT ELEVATION

Self-employed people and small businesses can use a micro to keep track of accounts and invoices. A freelance architect or designer could also use the graphics on a micro.

A small space like the front of a car needs to be designed so that the driver can reach all the controls and has enough room. A designer can buy a special program which draws people on the screen. They can be moved about to see if they fit into the design.
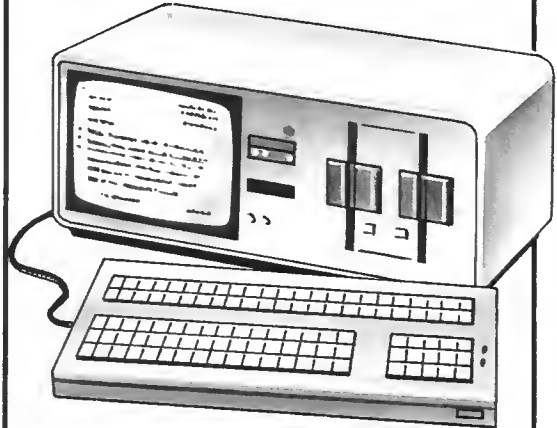
Microcomputer wordprocessors are used in offices to cut down on typing and paperwork. Standard letters and documents are typed and corrected on a wordprocessor and then stored on disk to be printed out when needed.

37

Adding to your micro

Once you are familiar with your micro and what it can do, there are lots of things you can buy to add to it. Extra equipment such as disk drives, printers, and graphics pads are called peripherals. To connect something to the micro you need an interface to convert the signals between the two, and different pieces of equipment need different interfaces. A micro usually has interfaces for a cassette recorder and TV built into it. Many also have interfaces for a printer, disk drive or light pen. If not, you can buy a separate one. Many peripherals, especially printers, plotters and modems (for telephone linkage to other computers) use a standard interface called an RS232. If you want to add several peripherals to the micro, you can buy a "motherboard" into which you can slot boards or cards containing interfaces for different equipment.

Extra memory

Add-on RAM pack

Some RAMs can be increased up to 64K, which means that you can put in over 2,500 program instructions.

You may want to increase your micro's RAM size before you buy any peripherals. You can then use longer programs for more exciting games and better graphics. You can buy add-on RAM packs for some micros. These are cartridges containing RAM chips which slot into the micro to connect with the PCB. Other micros have space on the PCB for extra RAM chips to be fitted by a dealer.

Disk drives

Sinclair microdrive
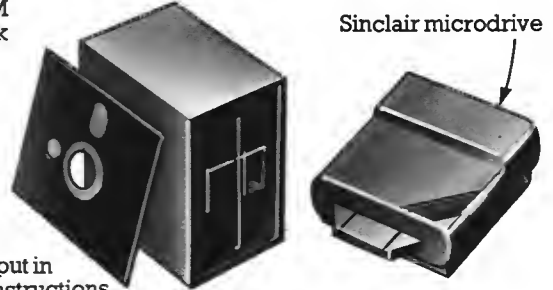
If you want to store long programs, or lots of information in a home database, you will find a disk drive much quicker than a cassette recorder. They are far more expensive, though. Disk drives for home micros usually use minifloppy disks which measure about 13.5cm across. The Sinclair microdrive uses even smaller disks, called microfloppies. They store 100K each, which is about enough space for all the words in this book.

Joysticks and paddles

Joystick   Fire button   Paddle
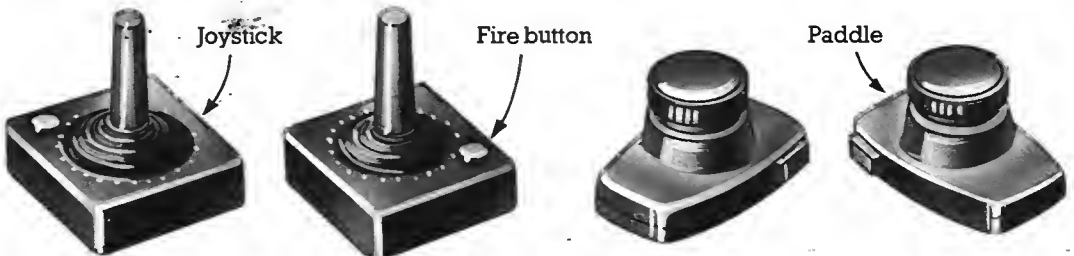
Joysticks and paddles are useful for playing arcade-type games where you want to move things like aircraft and spaceships around the screen. You can use keys on the keyboard for this, but joysticks and paddles give you more control and are more fun to use. With a joystick you can move the object in any direction, but paddles only move it up, down, left and right. Usually joysticks have a "fire" button on them to fire missiles. Many home micros have built-in interfaces for joysticks.

Graphics

If you are interested in graphics, you can produce exciting pictures on the screen by drawing on a graphics pad. You can also get good quality "hard copy" pictures using a plotter. A pen is supported over a sheet of paper and the movement of the pen is controlled by the computer's program. These are expensive pieces of equipment, though, so you could buy a light pen instead which is much cheaper.

Graphics pad

Light pen

Plotter

Some plotters can use several different coloured pens to draw pictures.
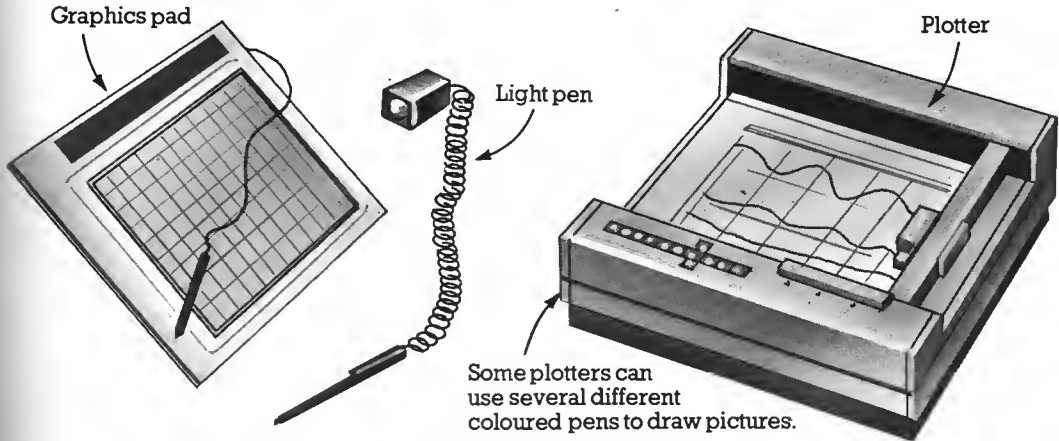
You can buy a high-resolution graphics card or cartridge for some micros to improve the picture quality. As well as providing more colours, it makes the groups of pixels you can control smaller, so details can be finer. It might make characters smaller, too, so you can fit more lines of text on the screen. High-resolution graphics use up a lot of memory, so you may need more RAM as well.

More about printers

There are three main types of printer – thermal, dot matrix and daisy-wheel. Thermal printers are the cheapest, and though the print can be rather messy they are adequate for printing out programs.

Dot matrix printers are also quite cheap. Daisy-wheel printers are expensive, but give very good quality print. Bi-directional printers print one line going forward and another going back to save time.

Thermal print head

Dot matrix print head

Pins

Letter shape

Ink ribbon

Daisy-wheel print head
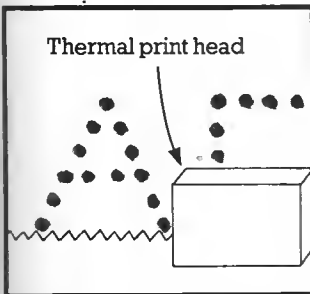
Thermal printers send out little sparks on to heat-sensitive paper, which turns black where a spark hits it. Patterns of black dots form letters.
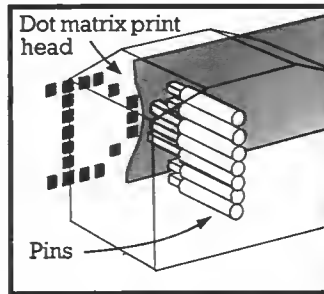
Dot matrix printers have a print head consisting of lots of pins. Letters are formed by combinations of pins shooting out and making dots on the paper.
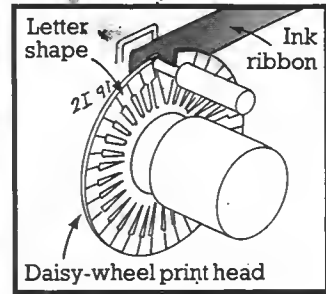
The print head on a daisy-wheel printer looks like a bicycle wheel without a rim. On the end of each "spoke" is a character shape.

Buyer's guide

On the next few pages there are descriptions of some of the main home computers currently available. They are arranged roughly in order of price with the least expensive models first.

If you are new to computers, the jargon used to describe them is very confusing at first. If you are buying your first computer, though, there are only a few main features you need be concerned with and these are outlined in the descriptions on the next few pages. At the bottom of this page there are some explanations to help you understand some of the terms used.

The best way to find out about the different computers is to ask friends who own computers about their machines or go along to a computer user's group and talk to the people there. You can also read the reviews in computer magazines and ask lots of questions in computer shops. (Ask the assistant to explain any terms you do not understand.) Before you buy your computer, think carefully about what you want to use it for and decide how much you want to spend. If you decide to buy a simple, inexpensive machine, check how much extra memory you can add and whether you can also use it with other equipment such as a disk drive or printer, etc. If you become a keen computer hobbyist, you will soon find you outgrow the simplest version of the computer and will want to add to it.

Processor This is the microprocessor, the CPU of the micro. The specifications for a micro usually tell you which processor the micro uses. The two main processors are the 6502 and the Z80 (see page 28). If you are buying your first micro you need not really worry about this.

Keyboard Most micros have a keyboard like an electric typewriter. A few, though, have touch sensitive keys which do not move when you press them. Typing in programs on a touch sensitive keyboard takes a bit longer than on one with keys which move.

Most micros have the same arrangement of letters as on a typewriter. This is called a QWERTY keyboard. (QWERTY is the sequence of letters in the first row of letters on a typewriter.) Sinclair computers (see opposite) have a special system where each key carries a programming word as well as a letter. This means you do not have to type in the programming words letter by letter.

Screen display The number of characters (i.e. letters and symbols) that the micro can display on the screen is measured in columns for the number of characters across the screen and lines for the number of lines of text which will fit down the screen. Some micros have automatic scrolling – when the screen is full the text automatically moves up the screen to make space at the bottom.

Graphics Picture quality is measured by the number of points you can plot across and down the screen. This is called screen resolution.

Interfaces Most micros have built-in interfaces for a TV and/or monitor and for a cassette recorder. They may also have interfaces for some of the following: printer, disk drive, joysticks, Prestel and for networking (linking up with other computers). If a micro does not have the interface you want you can usually buy one separately.

Software This is all the programs for a micro on cassette, disk or in printed form. The software for one micro does not work on another micro unless they are related machines like the Sinclair ZX81 and Spectrum.

ZX81 (Sinclair)

Z80A processor
1K RAM expandable to 16K
32 column × 24 line screen display
63 × 43 screen resolution

The ZX81 is a small, inexpensive microcomputer. It has a touch sensitive keyboard with the Sinclair keyword system – each key carries a programming word so you do not have to type in the words letter by letter. It uses a TV set for display and an ordinary cassette recorder for saving and loading programs. It can make only a black and white display. It also has a built-in interface for the Sinclair printer.

The ZX81 is the world's biggest selling microcomputer and there is probably more software for it than for any other computer. Most of the programs are games programs on cassette or printed in books and magazines.

ZX Spectrum (Sinclair)
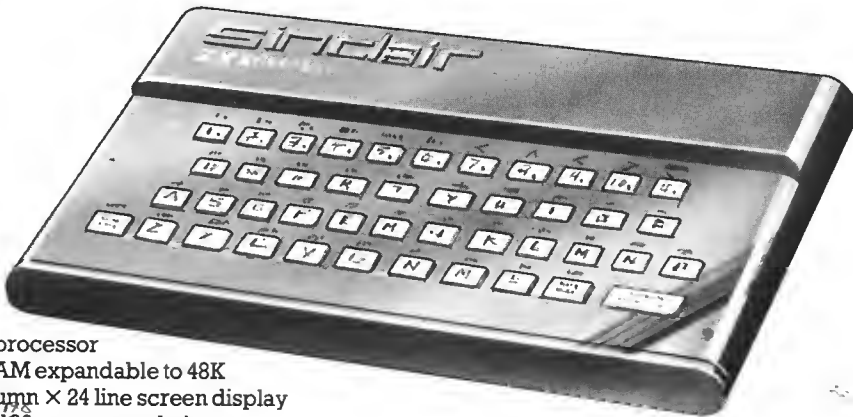
Z80A processor
16K RAM expandable to 48K
32 column × 24 line screen display
256 × 192 screen resolution

The ZX Spectrum is less than twice the price of the ZX81, but it has a much larger memory. It has the same keyword system for entering programming words as the ZX81, but its keys move when you press them. It can also make colour pictures and sounds.

It uses a colour TV set for display and an ordinary cassette recorder for saving and loading programs. It also has interfaces for the Sinclair printer and for a microdrive. This is a small disk drive for saving and loading programs on microfloppy disks. You can also add an RS232 and a networking interface to this micro.

It can make pictures with eight colours and sounds of over ten octaves. It has an internal loudspeaker but only one sound channel so you can only play one note at a time.

Most software produced for the ZX81 works on the Spectrum, but there are also lots of programs written specially for it.

41

PC1500 (Sharp)

This pocket sized microcomputer can be powered by batteries or the mains so you really can take it anywhere. It has its own built-in black and white liquid crystal display screen. To load and save programs on cassette you can use an ordinary cassette recorder, but you need a special interface unit which the computer fits into. This interface also works with the Sharp printer which can print in four colours. The small screen makes it not really suitable for games.

CMOS processor
3.5K RAM expandable to 7K
26 character display in one line
7×156 screen resolution
dimensions 20.5cm $\times$ 9cm
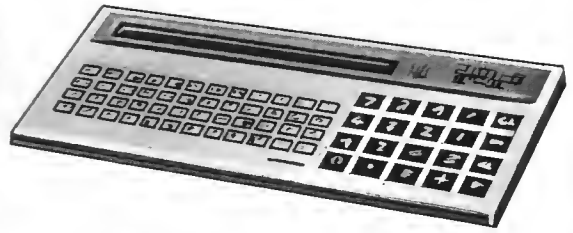
VIC 20 (Commodore)

6502 processor
5K RAM expandable to 29K
22 column $\times$ 23 line screen display
176×158 screen resolution

This is a small, sturdy home computer with colour graphics and sound. It uses a colour TV set for display, but needs a special VIC cassette recorder for storing and loading programs. There are lots of programs available on cassette and cartridge, and printed in magazines.

On the standard machine you can produce graphics in 16 colours using the symbols on the graphics keys. To produce graphics with DRAW and other BASIC graphics commands you need a special graphics program cartridge. For sounds the VIC uses the TV loudspeaker and can make four different sounds at the same time.

It also contains built-in interfaces for the following equipment: a disk drive, printer, joysticks, light pen and there is also an RS232 interface cartridge.

Electron (Acorn)

6502 processor
32K RAM
This is a new computer and some of the technical specifications were not released when this book was published.

The Electron is a small colour computer made by Acorn, the company who produce the BBC micro. It has a QWERTY keyboard with moving keys and uses the same version of BASIC as the BBC micro, so most BBC programs also work on the Electron.

TI-99/4 (Texas Instruments)

The TI-99/4 uses a TV set for display and can make pictures in 16 colours. It also has good music and sound effects, using its own internal loudspeaker. It can play three notes at once, over five octaves. You can also buy a separate speech synthesizer which can pronounce over 200 words.

The TI-99/4 uses an ordinary cassette recorder for saving and loading programs and there is lots of software on cassette, cartridge and disk. Other extra equipment for this micro includes a printer, disk drive, joysticks and an RS232 interface.
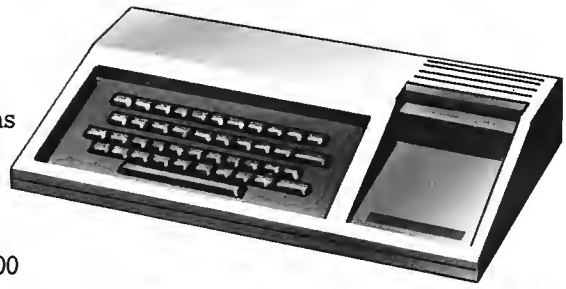
9900 processor
16K RAM expandable to 48K
29 column × 24 line screen display
256 × 192 screen resolution

Dragon (Dragon Data)

This is a small micro designed for home use. It has a keyboard with moving keys and uses a TV set for display and an ordinary cassette recorder. It has good colour graphics using nine colours, and can make a wide range of sounds using one sound channel and the TV loudspeaker. Additional equipment for the Dragon includes a printer, disk drive, joysticks, and RS232 and Prestel interfaces.

6809 processor
32K RAM expandable to 64K
32 column × 16 line screen display
256 × 192 screen resolution

Atari 400 (Atari)

6502 processor
16K RAM not expandable
40 column × 24 line screen display
320 × 192 screen resolution

The Atari 400 has a flat, touch-sensitive QWERTY keyboard. It uses a TV set for display, but needs its own cassette recorder for loading and saving programs from cassette. Much of the software for this machine, though, is in cartridges which slot straight into the micro. There is a wide range of good games programs for the Atari 400 and you can buy joysticks as an optional extra. Other additional equipment includes a disk drive and printer and the micro has 16 colours for graphics and four sound channels.

TRS-80 Colour Computer (Tandy, or Radio Shack in the U.S.A.)

The TRS-80 Colour Computer uses a colour TV set for display and an ordinary cassette recorder for saving and loading programs. It has eight colours and can also make sounds. Additional equipment for the Colour Computer includes joysticks, a printer, a disk drive and an RS232 interface.

Program cartridge

6809E processor
16K expandable to 32K
32 column × 16 line screen display
256 × 192 screen resolution

Atom (Acorn)

The Atom is a small hobby or home business computer. It uses a TV set for black and white display, but you can buy an extra PCB to make colour graphics. It uses an ordinary cassette recorder. Additional equipment for the Atom includes a disk drive, printer, joysticks and a network and Prestel interface. There is a good supply of varied software on cassette, disk and printed in books and magazines.

6502 processor
2K RAM expandable to 12K
32 column × 16 line screen display
256 × 192 screen resolution

BBC Micro (Acorn)
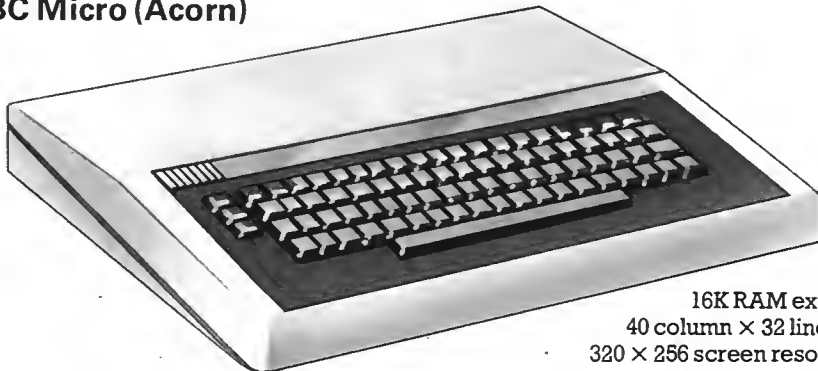
6502 processor
16K RAM expandable to 32K
40 column × 32 line screen display
320 × 256 screen resolution (Model A)

The BBC micro comes in two versions, the Model A which is the basic version and Model B, a more advanced system. The Model A can be upgraded to Model B level and the complete system with all the available peripherals makes a powerful small computer.

It uses an ordinary TV set and cassette recorder and has eight colours and three sound channels. Additional equipment available for the BBC micro includes a disk drive interface, network interface, speech synthesizer, program cartridge interface, paddles, printer and Prestel interface. There is extensive software of all kinds.

44

PET (Commodore)

6502 processor
16K RAM expandable to 96K
40 column × 25 line screen display
512 × 512 screen resolution

The PET was one of the first "personal computers", that is, a computer designed to be used by one person. It has a built in monochrome screen and needs a Commodore cassette recorder which may also be built into the micro.

The PET is used mainly by serious computer hobbyists and by businesses and schools. Additional equipment for the PET includes a disk drive and printer and it has a built-in PET IEEE-488 interface (this is an alternative to the RS232 interface). There is an extensive range of educational, business and games software on cassette or disk.

Apple II Plus (Apple)

6502 processor
16K expandable to 48K
40 column × 24 line screen display
280 × 192 screen resolution

The Apple, like the PET, is used by serious computer hobbyists, small businesses and schools.

The basic machine has a motherboard with a number of slots into which you can fit PCBs for various different functions, e.g. for memory expansion and PCBs which enable you to use other programming languages.

Other additional equipment includes a printer, graphics tablet, disk drive and Prestel interface. There is a lot of educational, business and games software for the Apple, in printed form and on disk.

45

Here are some additional computers arranged in order of price starting with the least expensive.

Jupiter Ace (Jupiter Cantab)

Z80A processor
3K RAM
32 column × 24 line screen display

This small, inexpensive microcomputer uses the programming language FORTH instead of BASIC. It has a calculator type keyboard, good graphics and sound, but no colour. It is the first home computer to use FORTH so there is very little software and most of the books on how to program in FORTH are quite complex. It has a printer interface and a microfloppy disk drive.

Oric 1 (Oric Products International)

6502A processor
16K RAM
40 column × 28 line screen display
240×200 screen resolution

Inexpensive microcomputer with a calculator type keyboard, sixteen colours and four sound channels. It has a Centronics printer interface. A more expensive version of the same computer is available with 48K RAM.

Newbrain (Grundy)

Z80A processor
32K RAM expandable to 2Mbytes
40 or 80 column × 25 line screen display
640×250 screen resolution

There are two versions of the Newbrain. The more expensive version has a single line fluorescent display built into the keyboard, but it can be used with a TV or monitor, too.

One of the main features of this computer is the extent to which the memory can be expanded by adding plug-in memory modules. It has no colour or sound facility, but the black and white graphics are very clear. It is designed for expansion and has a number of sockets for attaching a printer, modem and expansion board and networking interface. There is a variety of mainly business orientated software available in ROM cartridges.

Colour Genie (EACA)

Z80 processor
16K RAM expandable to 32K
40 column × 24 line screen display
196×96 screen resolution

The Colour Genie has many similarities to the Tandy computers. It can run Tandy software listings with minor alterations, but not the cassettes. It has a typewriter type keyboard and printer, disk drive, light pen, joystick and RS232 interfaces.

Lynx (Camputers)

Z80A processor
48K RAM expandable to 192K
40 column × 24 line screen display
248×256 screen resolution

This micro has a typewriter type keyboard and eight colour graphics and sound. It is an expandable system with a disk drive, printer and other add-ons.

Commodore 64 (Commodore)

6502 processor
64K RAM
40 column × 25 line screen display
320×200 screen resolution

The Commodore 64 has the same keyboard layout as the VIC 20, but a much larger memory and a standard size screen display. It has good graphics with sixteen colours and three sound channels. The 64 can use most of the PET and VIC software and also has a good range of business and games cartridges of its own. There is a disk drive and modem interface and a plug-in cartridge which makes it compatible with PET peripherals.

HX-20 (Epson)

6301 processor
16K RAM expandable to 32K
20 column × 4 line screen display
120×32 screen resolution

This micro has a small, neat keyboard with a built-in screen, printer and micro-cassette recorder. It can also be connected to a TV or monitor and can work off batteries or the mains. Additional equipment for the HX-20 includes ROM cartridges, a barcode reader, acoustic coupler for linking to another computer via the telephone, and an RS232 interface.

Micro words

Animations   Moving pictures on the screen.

Arithmetic and logic unit (ALU)   The circuits in the central processing unit where calculations and comparisons are carried out.

ASCII   American Standard Code for Information Interchange. A standard way of representing letters and numbers with eight-bit binary numbers.

Backing store   Programs or data saved outside the computer on tape or disk.

BASIC   A general-purpose programming language suitable for most kinds of programs. The letters stand for Beginners' All Purpose Symbolic Instruction Code. Most micros use BASIC.

Baud rate   A measurement of the speed at which one bit travels from one part of the computer to another, or between a computer and a peripheral, for instance a cassette recorder. One baud is one bit per second.

Binary   A counting system using only two digits (0 and 1). Machine code is a binary code.

Bit   One of the two digits (1 and 0) that make up binary code. In a computer, a bit is a pulse signal (1) or a no-pulse signal (0).

Bug   A mistake in a program.

Bus   Tracks along which data is moved about the computer.

Byte   Most micros work with groups of eight bits at a time, called a byte.

Central processing unit (CPU)   The circuits which control all the other parts of the computer and where calculations are carried out.

Character   A number, letter or symbol.

Chip   A tiny slice of silicon with lots of electronic circuits etched into it. They are kept in protective cases. Chips are used in computers to do all the work.

Compatibility   Computers are said to be compatible if they can understand the same programs.

Data   Any information you give the computer which will be worked on according to the instructions in a program. The information and results from a computer are also called data.

Database   An organised file of information held in the computer's memory or on tape or disk.

De-bugging   Finding mistakes in a program and correcting them.

Dialect   There are several versions of BASIC, called dialects, which use slightly different commands.

Error message   A message the computer flashes up on the screen to tell you that there is a bug in the program, and sometimes what kind of bug it is and where it is.

Fortran   A high-level programming language used mainly by scientists and mathematicians.

Gate   An arrangement of transistors which works on the pulses travelling through the circuits of a computer. All the computer's processing is done using gates.

Graphics   Pictures made with a computer.

Hard copy   Programs or data printed out by the computer using a printer.

Hardware   A computer, or a piece of related equipment, such as a disk drive or printer.

Hex   A counting system based on 16 digits (0 to 9 and A to F). It is useful for low-level programming as an eight-bit byte can be expressed as two hex digits.

Input   Any information or instructions you feed into the computer.

Integrated circuit (IC)   Minute electrical circuits containing thousands of electronic components on a tiny chip of silicon.

Interface   Special circuits which convert the signals from a computer into a form other electronic equipment can deal with, and vice versa. Different pieces of equipment need different interfaces.

Interpreter   A special part of the computer's permanent memory (ROM) where instructions in a programming language (usually BASIC in a micro) are converted into machine code.

Kilobyte (K)   One kilobyte is 1024 bytes.

Listing   A program written, typed or printed out on paper.

Load   Put a program into a computer's memory from cassette tape or disk.

Machine code   The pattern of electronic pulse signals which the computer uses to do all its work.

Microprocessor   A chip containing all the different kinds of circuitry a computer needs to control an electronic device. The CPU of a microcomputer is a microprocessor chip.

Mnemonics   A code consisting of abbreviated instructions. Mnemonics are used as an aid to low-level programming.

Modem   Short for modulator/demodulator. A device which converts the signals from a computer into a form which can travel down telephone lines.

Monitor   Part of the ROM which holds instructions telling the CPU how to operate.

Motherboard   A circuit board into which you can slot other PCBs.

Network   A system of computers, sometimes with other computer peripherals, linked together to share information.

Output   Any information the computer gives you.

Pascal   A high-level programming language for general use.

PCB   See Printed circuit board.

Peripherals   Equipment that you can attach to a computer, such as extra screens, printers or plotters.

Pixels   Tiny areas on the screen which the computer can switch off or on to make the shapes for letters or pictures.

Port   A socket on a micro where you plug in a lead connecting it to another piece of equipment.

Printed circuit board (PCB)   The board inside a computer which holds all the chips and other components. It has metal tracks on it to carry the electrical signals between the components.

Program   A numbered list of instructions to make the computer do a particular job.

Programming language   A language in which a program of instructions for a computer must be written. There are lots of different languages. High-level languages consist of words and symbols and are easier to use than low-level languages which resemble machine code more closely.

Random access memory (RAM)   Chips where any information you give the computer is stored. You can retrieve or change this information. The RAM chips empty of stored information every time the computer is switched off.

Read only memory (ROM)   All computers have ROM chips which store instructions telling them how to work. The instructions are built into the ROM chips when they are made, and the information in ROM is permanent.

Save   Store a program outside the computer, usually on tape or disk.

Screen resolution   The number of pixel groups on the screen which the computer can control. High resolution graphics are detailed pictures made by a computer which can control lots of small groups of pixels. Low resolution graphics are pictures made with fewer, larger groups of pixels.

Sensor   A device outside a computer that measures light, pressure or temperature and sends information back to the computer.

Software   Computer programs.

Syntax error   A bug in a program due to a mistake in the programming language.

Synthesizer   A piece of equipment or circuitry which produces musical notes or sounds through a loudspeaker.

System variables   An area of RAM which stores information about different parts of the computer, for instance, where the next character will be printed on the screen, and the addresses of the boundaries between different areas of RAM. These can shift depending on how much is stored in each area.

Transistor   An electronic component which stops or sends on the pulses in the circuits of a computer, depending on the pulses it receives. A single chip contains thousands of transistors.

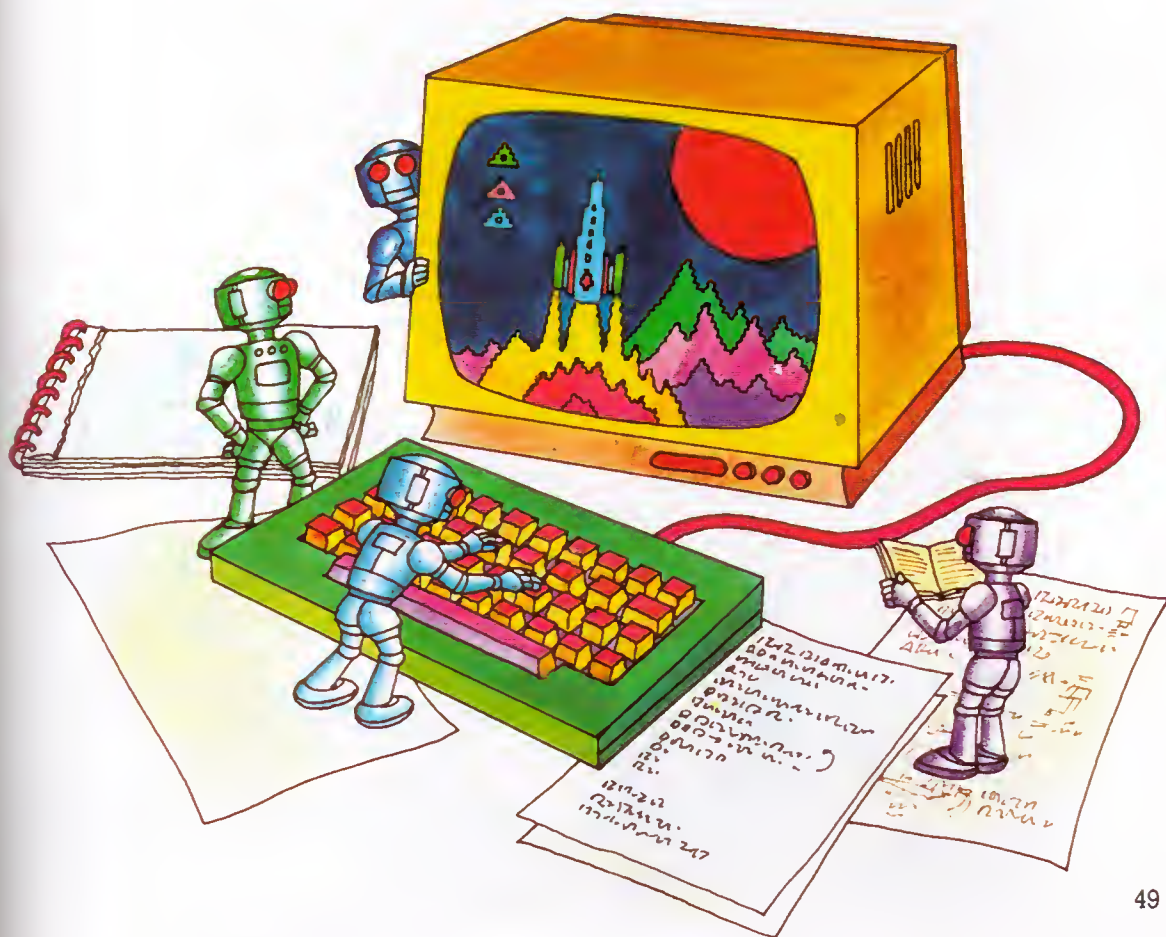**Visual display unit (VDU)**   A screen, similar to a TV screen, designed specially for a computer.

INTRODUCTION TO
COMPUTER PROGRAMMING

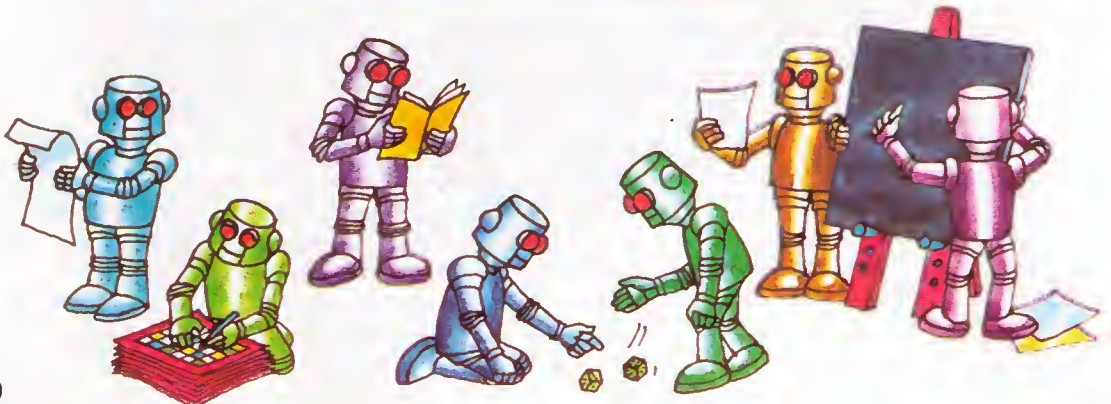## Brian Reffin Smith

Edited by Lisa Watts

Designed by Kim Blundell
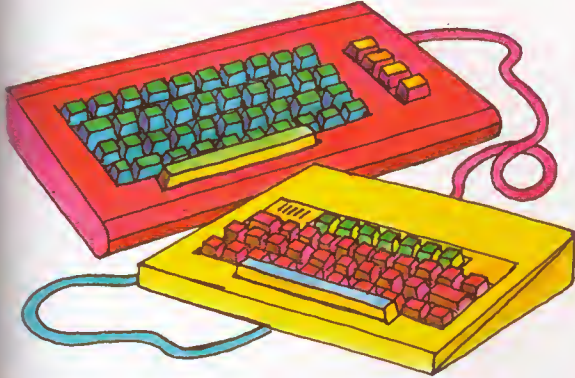
### Illustrated by Graham Round and Martin Newton

Contents

This section is a guide to writing computer programs in BASIC for absolute beginners. BASIC is the language used on most home computers. It is a way of writing instructions for a computer in a form the computer can understand.

You do not need a computer to use this guide, though of course it helps you to understand the programs if you can try them out on a computer. Different makes of computer use slightly different versions of BASIC. Nearly all the terms in this book, though, will work on most microcomputers, and the few that are not standard are clearly marked.
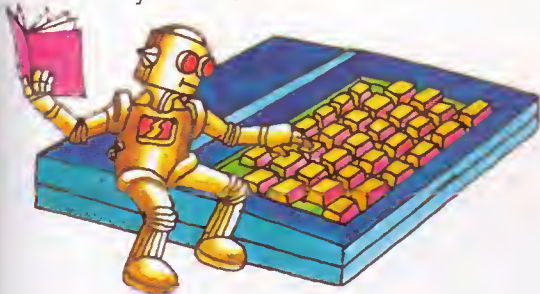
At the beginning there are some guidelines to programming a computer. Then, as you read through the book, the main BASIC words are introduced one by one, with short programs to show how they work.
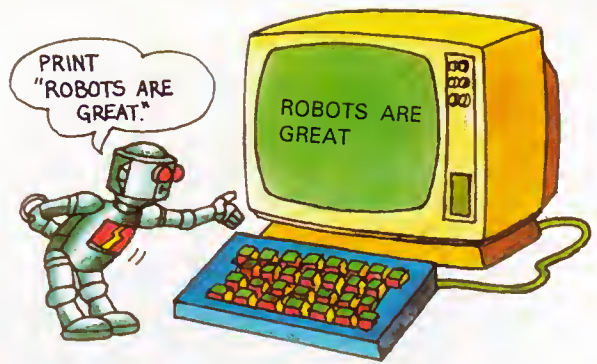
To give you some practice in writing programs there are program puzzles to solve and suggestions for programs to write and for useful alterations you can make to the programs in this guide. The answers to the program puzzles are on pages 92-93.

At the end of this section there are some guidelines to help you write programs, and a list of "bugs" – the mistakes in programs which stop them working – with hints to help you recognize them. At the end of the book there is a summary of BASIC with a short explanation for each of the terms.
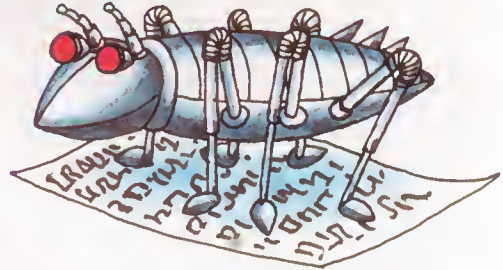
If you have a micro, try out the programs in this guide, then, to find out more about how your micro works, look up the BASIC terms in your manual. You may find that some of the rules given here are not necessary on your micro. The best way to learn BASIC is to try out lots of programs from books and magazines, then alter them a little to see what happens. From there you will soon be writing your own programs.

How a computer works

You can do all kinds of different things with a computer. You can draw pictures, write funny poetry, draw graphs, play games and do lots of things with words and numbers, some of them useful and some just for fun. A computer is sometimes described as an "information processor". Its task is to take the information you give it, work on it according to your instructions, and show you the results.

DIVIDE 999
BY 27

PRINT MY NAME
80 TIMES

DRAW 1000,100

LOAD
"MISSILE
COMMAND"

POKE
36876,15
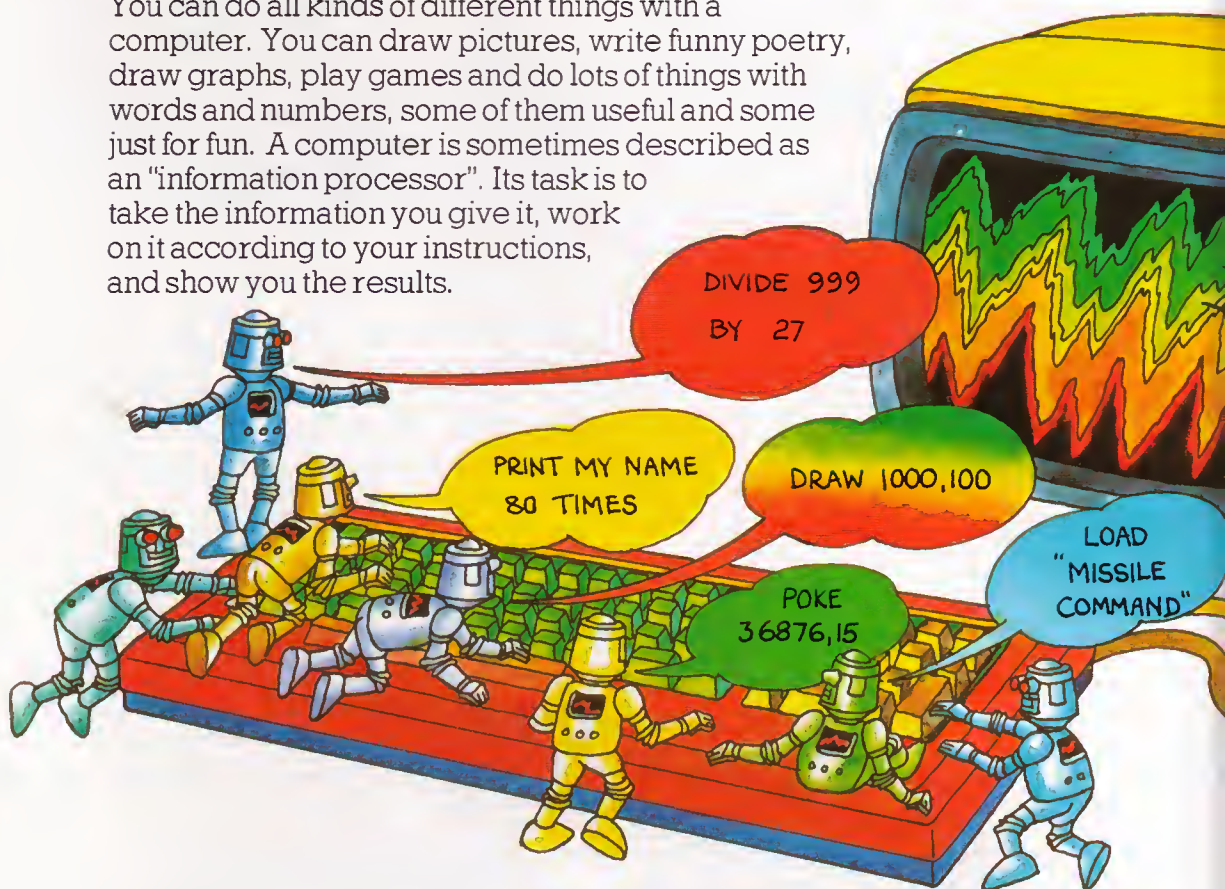
To make a computer do what you want you have to give it very precise instructions. A list of instructions for a computer is called a program* and the information you give the computer to work on is called data. The program has to be written in a language, such as BASIC, that the computer can understand, and it must follow all the rules of the language too.

Microcomputers

Most micros consist of a keyboard which you plug into a TV set. You give the micro instructions and information by typing on the keyboard and everything you type, along with the computer's results, is displayed on the TV screen.

Some micros have a small, built-in display screen, like a pocket calculator. A few use a special screen called a monitor. A monitor is like a TV but it cannot pick up the signals from TV stations.

The keyboard of a micro looks like a typewriter keyboard with some extra keys. On some micros each key gives the computer a separate instruction in BASIC so you do not have to type the words in letter by letter.

52

*Spelt like this when used for a computer.
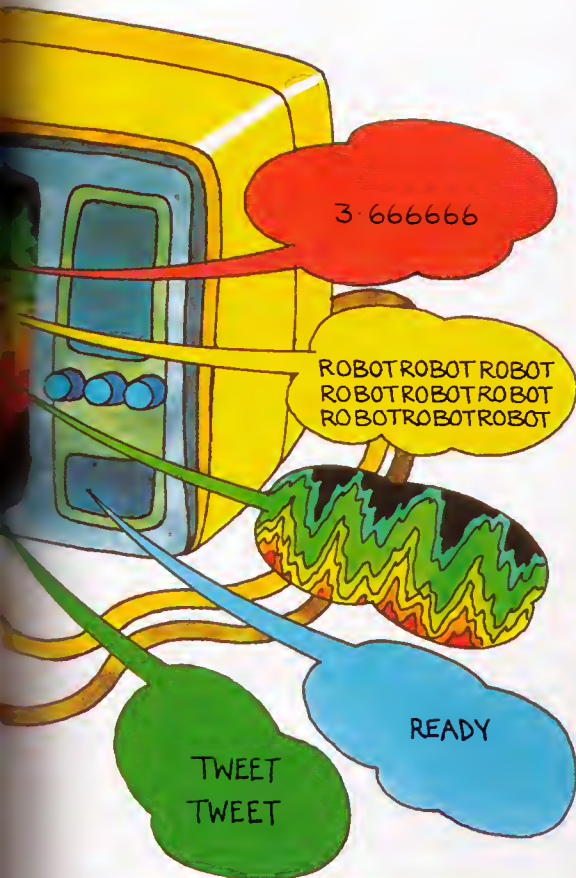
Inside a micro

A micro is made up of two main parts: the central processing unit (CPU) where all the work is done, and the memory where programs and data are stored.

CPU

ROM

RAM

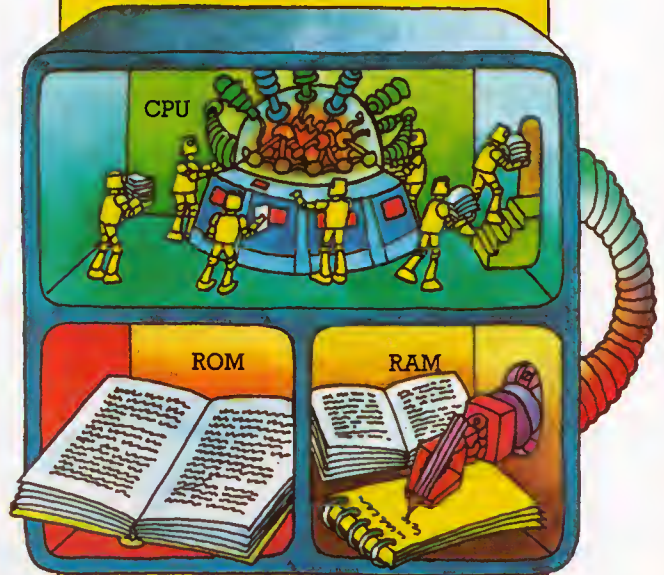In fact, the computer has two memories. One, called ROM, contains a program which controls all the operations of the computer. The other, called RAM, is an empty memory where your programs and data are stored. When you switch off the micro all the information in RAM is lost, but the ROM program is permanent.

3·666666

ROBOT ROBOT ROBOT
ROBOT ROBOT ROBOT
ROBOT ROBOT ROBOT

READY

TWEET
TWEET
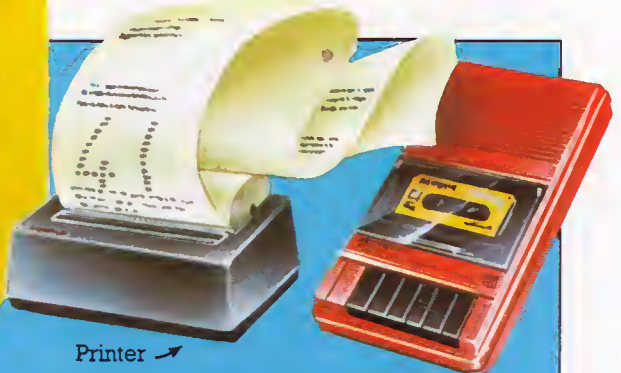
Printer

A TV screen is the most usual way to display the information from a micro. You can also print it out on paper, using a printer. This is useful as the information in the micro and on the TV screen is lost when you switch them off.

Another way to store information from a micro is with a cassette recorder. You can store programs and data on a cassette, then load them back into the micro when you want to use them.

Giving a computer instructions

1

PRINT "ROBOTS ARE GREAT"

PRINT "ROBOTS ARE GREAT"
ROBOTS ARE GREAT
READY

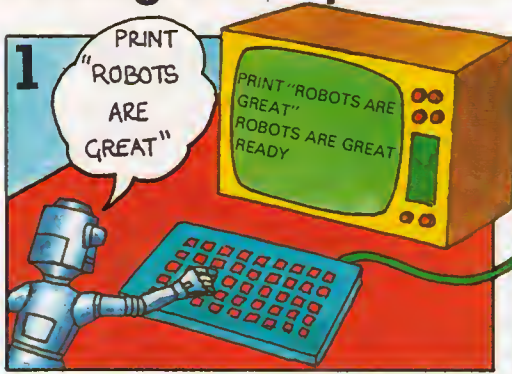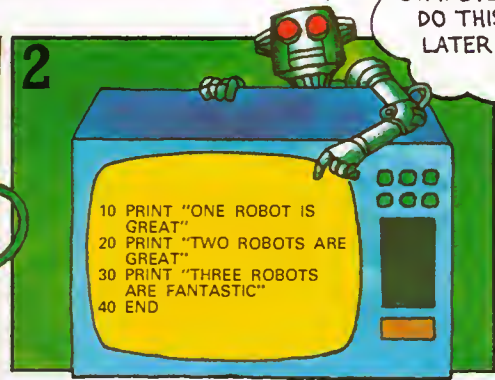To make the computer do something, you have to type in an instruction it understands. This instruction can be a direct command which it carries out

2

I WANT THE COMPUTER TO DO THIS LATER ON

10 PRINT "ONE ROBOT IS GREAT"
20 PRINT "TWO ROBOTS ARE GREAT"
30 PRINT "THREE ROBOTS ARE FANTASTIC"
40 END

straight away, or it can be a program of instructions which it stores in its memory and does not carry out until you give it the go-ahead.

3

PUNCTURE PROGRAM

1 REMOVE WHEEL
2 FIND TYRE LEVER
3 REMOVE TYRE.
4 MEND PUNCTURE
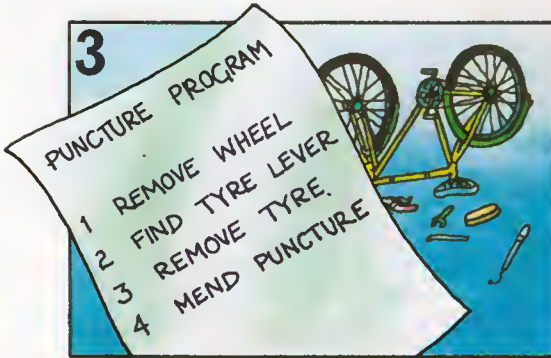
The instructions in a program have to be very carefully worked out. The computer will attempt to carry out your instructions precisely, even if they are wrong.

4

???!

??!?

PUNCTURE PROGRAM
1 REMOVE WHEEL
FIND TYRE LEVER
REMOVE TYRE
MEND PUNCTURE
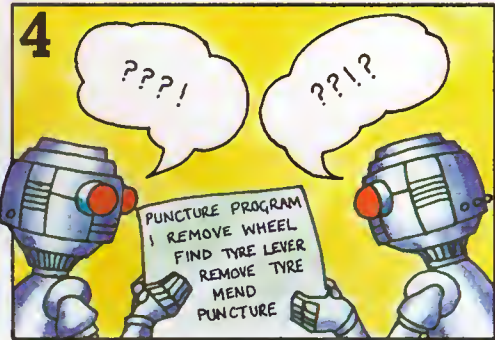
The computer cannot understand instructions written in our language, so you have to write them in one of the many computer languages. Some of these languages are described opposite.

5

11011101111001
10001101011011
11001011101011

PUNCTURE PROGRAM
1 ...
2 ...
3 ...
4 ...
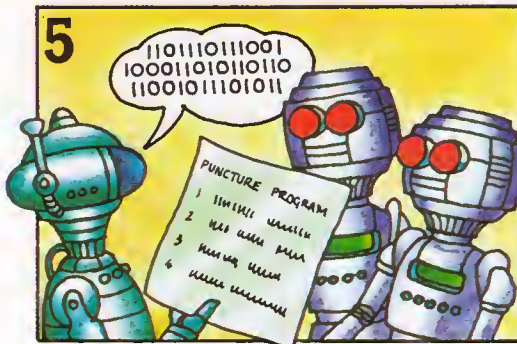
All the work inside the computer is done with a code of tiny pulses of electricity. Your instructions are translated into computer code by a special program inside the computer called the interpreter.

6

11001101101
11101001011100
1010100111
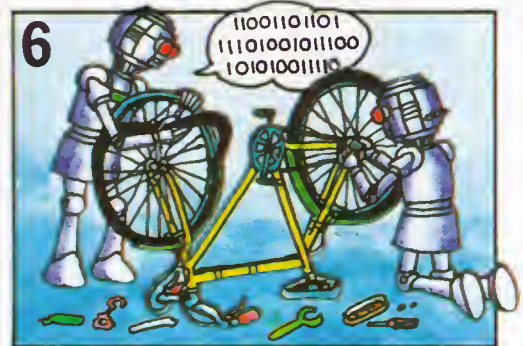
Each piece of information in computer code is represented by patterns of pulses. Computer code can be written down using 1 to represent a pulse and 0 to show there is no pulse.

54

Computer languages

You could write programs in computer code but it would be very difficult. Instead, there are special computer languages, called high level languages, which the computer can translate into its own code.

There are hundreds of different high level languages, many of them specially designed to do one particular kind of work. BASIC is one of the most common languages. The letters stand for Beginner's All-purpose Symbolic Instruction Code. It is not just used by beginners though. Below there are examples of three different languages.

Names program
```
10 PRINT "WHAT IS
      YOUR NAME"
20 INPUT N$
30 IF N$="BRIAN"
      THEN PRINT "HELLO"
40 IF N$="GEORGE"
      THEN PRINT "GO AWAY"
50 END
```

Money program
```
PROCEDURE MONEY;
VAR MONTH:INTEGER
BEGIN (*CALCULATE*)
  FOR MONTH:=1 TO
  TOTALMONTHS DO
    AMOUNTDUE:=
    (AMOUNTDUE:=
    (1 + MONTHLYINTERESTRATE)*
END; (*MONEY*)
```

Geography program
```
T: WHAT IS THE CAPITAL OF FINLAND
*INPUT A: $A$
T(LEN(A$)=0):GO ON, HAVE A GUESS
JC: @A
M: COPE!STOCK!OSLO!GOTHEN!LAP
TY: CLOSE BUT NOT CLOSE ENOUGH – TRY
      AGAIN
JY: @A
M: HELSINKI! HELSING
TY: GOOD – IT IS HELSINKI.
```

This is a short program in BASIC. Line 10 tells the computer to print "What is your name" on the screen. Then the computer stores your answer in its memory and if your name is Brian or George, it prints out a message to you.

This program is written in Pascal, a language named after a famous French mathematician. It is part of a program to work out details about money. Many people think it is easier to write good, neat programs in Pascal than in BASIC.

This is a language called PILOT. It is used to write programs to help people learn new subjects. In this language, the computer can recognize answers even if they are not exactly right.

$$\frac{\sin A}{a} = \frac{\sin B}{b}$$

$$5^2 = 3^2 + 4^2$$

$$\cos A = \frac{21}{24}$$

$$= 0.8750$$
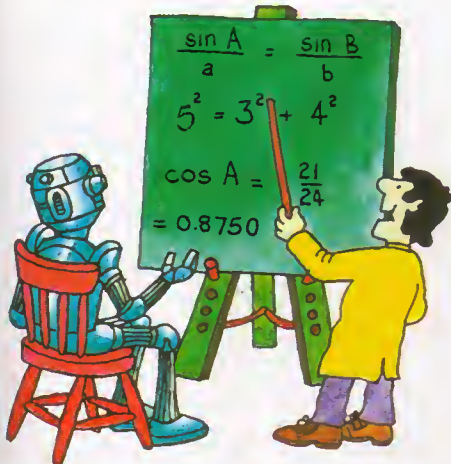
11. Bb3, Ne5
12. 0-0-0, Nc4
13. Bxc4, Rxc4
14. h5, Nxh5

TAITAA OLLA VIISITOISTA ASTETTA PAKKASTA*

HOW COLD IS IT TODAY ?

At first glance, computer languages seem very strange and difficult, but then, so do other languages such as the Finnish shown on the right, until you get to know them. There are lots of other subjects too, in which special languages are used. For instance, in mathematics a special notation is used to write down ideas and formulae which would need a lot of ordinary words to explain them and other kinds of notation are used to write down chess moves or music.

55

*Minus fifteen I guess.

Writing programs

A program is like the rules for a game, or the recipe for a cake. If there is a mistake in the rules, or the recipe, you will not be able to play the game properly, or bake a good cake. In the same way, the results you get from a computer depend on the instructions you give it. To write a program for a computer you first need to study what you want to do very carefully and work out the main steps needed to achieve the result you want.

Letter program
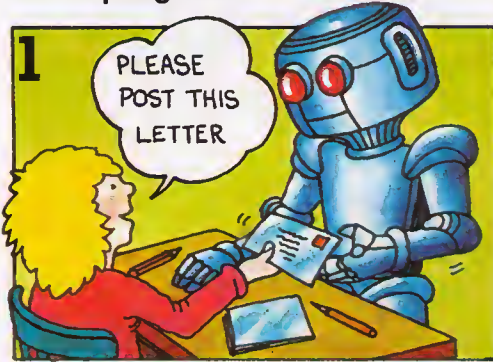
Imagine trying to write a program to tell a robot to post a letter. A simple instruction as shown above would be too difficult for the robot's computer brain to understand.
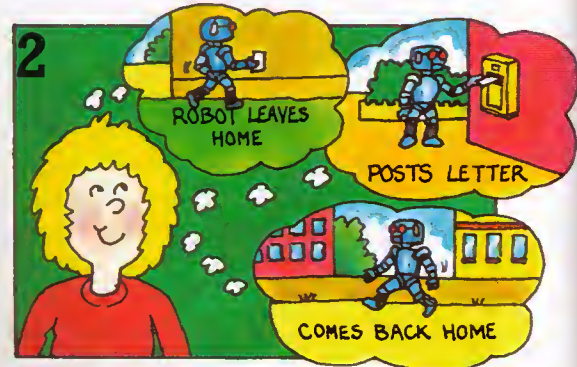
You need to work out exactly what the robot needs to do to post the letter. Its computer needs instructions telling it what to do at every stage.

Leave home
   Walk to front door
   Open door
   Go out and close door
   Find postbox

Post letter
   Hold letter in hole
   Let go of letter

Come home
   Turn round
   Retrace steps
   Open door
   Come in
   Close door

To write the program you need to break down the instructions for each stage into even smaller steps which can be translated into a language the robot can understand.
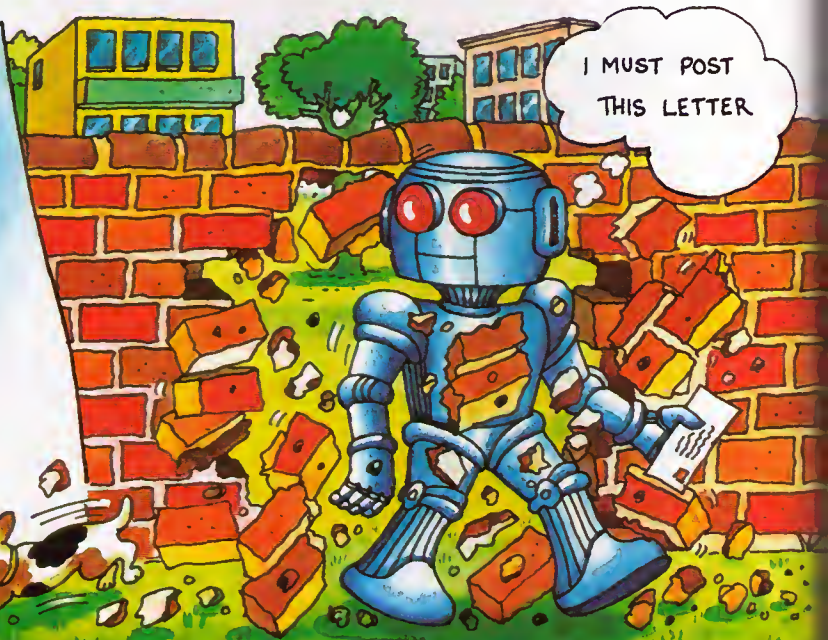
The robot will attempt to follow your instructions even if they are wrong, or incomplete. Mistakes in a program are called bugs and they can sometimes lead to unusual results from the computer.

Program diagrams

When you are writing a program it sometimes helps to draw a diagram like the one below, showing the main steps you need to solve the problem. A diagram like this is called a flowchart. It shows each of the steps the computer needs to carry out, and the order they should come in.

Robots rule OK flowchart

Start

Are you a robot? → Wait for next person

Answer yes or no

No

Yes

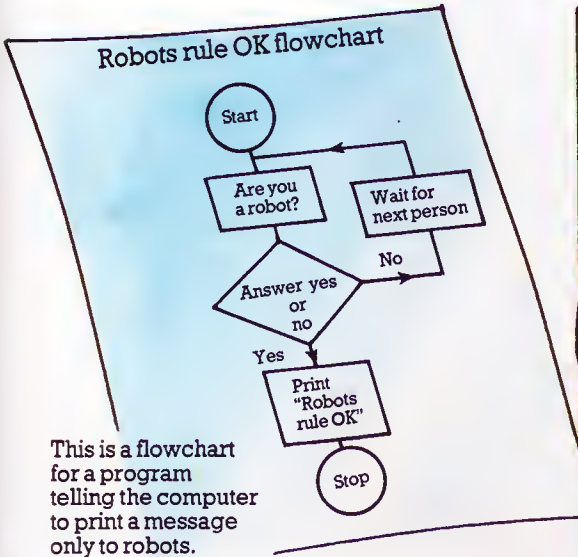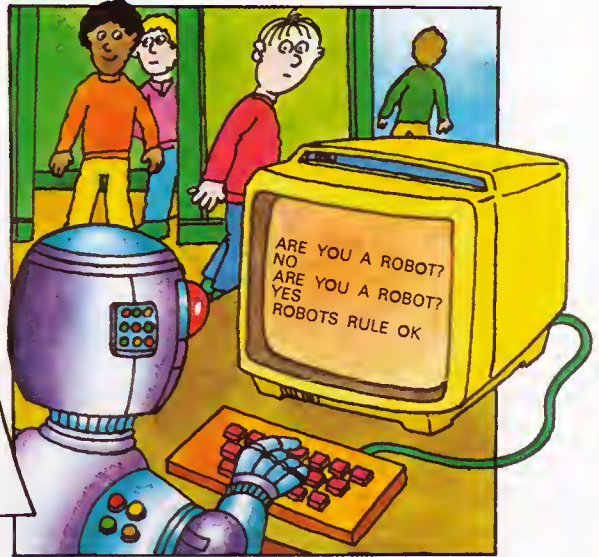Print "Robots rule OK"

Stop

This is a flowchart for a program telling the computer to print a message only to robots.

ARE YOU A ROBOT?
NO
ARE YOU A ROBOT?
YES
ROBOTS RULE OK

A flowchart has different shaped boxes for different steps in the program. The beginning and end of the program have round boxes, instructions telling the computer to do something are in rectangular boxes and decision boxes, where the computer can do different things depending on the information it receives, are in diamond-shaped boxes. The lines show the possible routes the computer can follow.
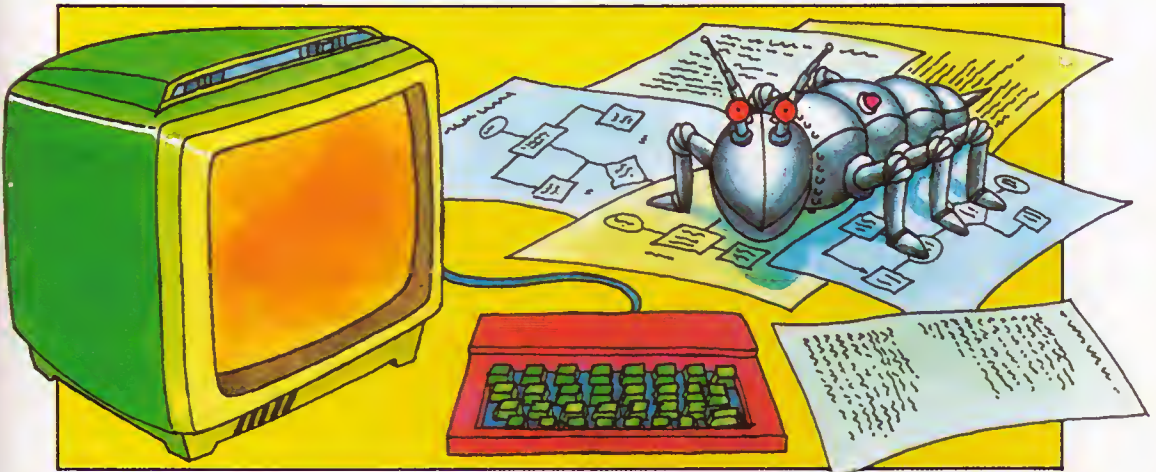
After working out all the details of the program you can translate it into BASIC and test it on the computer. The program will probably not work straight away though, as there will probably be some bugs in it. These may be typing mistakes made when you typed the program into the computer, or errors of logic in your program. Before you can get the program to work you have to find all the bugs and correct them.* Sometimes, a bug makes a program produce a slightly different result which you may prefer. Useful bugs like this are called "pugs".

57

*There are some tips to help you find bugs on pages 90-91.

First words in BASIC

Lots of the words in BASIC are based on English words and it is quite easy to guess what they mean. For instance, PRINT means "display on t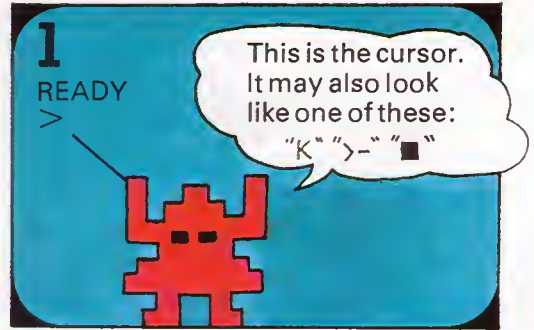he screen", RUN means "carry out this program" and INPUT means "give the computer information". On these two pages you can find out how to use the word PRINT.

Most home computers have a BASIC language interpreter inside them already and when you switch them on they are ready to be programmed in BASIC.*

1

READY
>

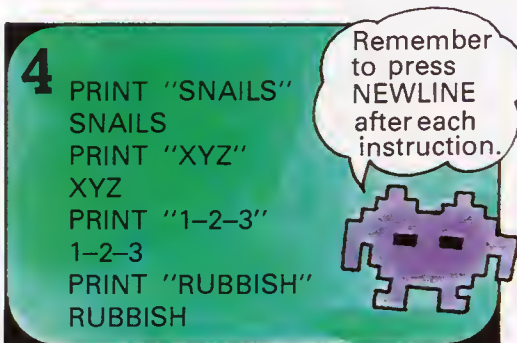This is the cursor. It may also look like one of these: "K" ">-" "■"

When you switch on a micro some words are usually displayed on the screen automatically, along with a small symbol called the cursor. The cursor shows where the next letter you type will appear.

2

READY
PRINT "SNAILS">

To tell the computer to display words on the screen you use PRINT with the words you want in quotation marks. For instance, PRINT "SNAILS" tells it to display the word SNAILS on the screen.

3

READY
PRINT "SNAILS"
SNAILS
>

Press NEWLINE, RETURN or ENTER to tell the computer to carry out an instruction.

The computer will not carry out your instruction, though, until you press NEWLINE (or RETURN or ENTER – it varies on different computers) to tell it the instruction is complete.

4

PRINT "SNAILS"
SNAILS
PRINT "XYZ"
XYZ
PRINT "1–2–3"
1–2–3
PRINT "RUBBISH"
RUBBISH

Remember to press NEWLINE after each instruction.

The computer will display on the screen whatever you type between the quotation marks. It can be letters, numbers, words or symbols. Note that it does not display the quotation marks themselves.

5

PRINT "1–2–3"
1–2–3
PRINT "RUBBISH"
RUBBISH
PRINT 999
999
PRINT 123
123

To clear the screen you type CLS, or the computer's own word.
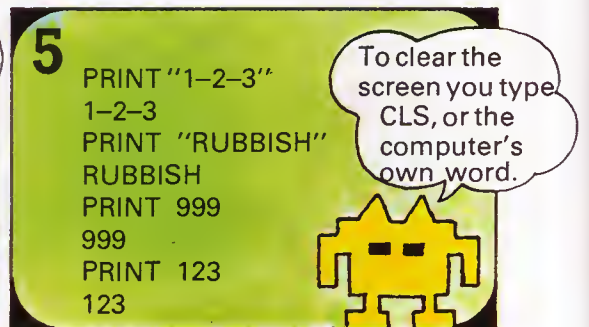
To display numbers by themselves, you do not need to use quotation marks. Now, to clear the screen you type CLS on most micros. (Check this in your manual if you have a computer.)

58

*Some computers have to have a special program loaded from cassette tape before they understand BASIC.

A program in BASIC

In a program, each line of instructions starts with a number. This tells the computer to store the instructions in its memory and not to carry them out until you give the go-ahead. On the opposite page, the instructions to the computer did not have numbers, so the computer carried them out straight away. Here is a short program which makes the computer display symbols in the shape of a face on the screen.

On some computers the figure 0 has a line through it, like this.

```
10  PRINT "//////"
20  PRINT "I    I"
30  PRINT "I (..) I"
40  PRINT "I -L I"
50  PRINT "VVVV"
60  END
```

Line numbers usually go up in tens so you can add extra instructions without renumbering the whole program.

Many computers do not need this line.

When you type in a program you have to press NEWLINE (or the computer's word) at the end of each line. The lines are displayed on the screen but the computer does not carry out the instructions until you tell it to by typing RUN. Be careful not to mix up the letter O and the figure 0 as this will cause a bug. Most computers have a RUBOUT or DELETE key for correcting typing mistakes.

```
RUN
//////
I    I
I(..)I
I - L I
VVVV
```

The computer displays exactly what you typed between the quotation marks, including the spaces.

When you have typed in all the lines, check them carefuly to make sure there are no mistakes. Then, to tell the computer to carry out the program, you type RUN, followed by NEWLINE.

```
RUN
MISSING"
LIST
```

Error message

You type LIST (and NEWLINE) to see program again.

If the program does not work, or the picture does not look right, you need to display the program again to find the bug. To do this you type LIST. The computer may give you an error message telling you what the bug is.

1 Debugging programs

```
RUN
MISSING"
LIST
10  PRINT "//////"
20  PRINT "I    I"
30  PRINT "I (..) I"
40  PRINT "I -L I"
50  PRINT " VVVV          ← No quotes
60  END
```

The computer will give you an error message for most bugs. The error messages are explained in the computer's manual. The easiest way to correct a mistake is to type the whole line again. The computer will replace the old line with the new one. To get rid of a line from

2

```
RUN
MISSING"
LIST
10  PRINT "//////"
20  PRINT "I    I"
30  PRINT "I (..) I"
40  PRINT "I -L I"
50  PRINT " VVVV
60  END
50  PRINT " VVVV "
```

Line typed in again correctly.

the program altogether, just type the line number, followed by NEWLINE. Each computer also has its own way for correcting or altering parts of lines, using words such as EDIT or COPY. This is explained in the computer's manual.

Program puzzle – Try changing the program to give the face different features.

Giving the computer information

To make the computer do something more useful than just displaying things on the screen you have to give it information or "data" to work on. The computer stores this information in its memory until you tell it to use it.

1

```
10 LET A=6
20 LET B=7
30 LET C=23
40 LET D=4
```

These are the numbers to be stored in the computer's memory.

These are the labels for the memory spaces.

When you put a piece of data into the computer's memory you have to give it a label so you can find it again. You can use letters of the alphabet as labels. To label a memory space and put a number in it you use the word LET, as shown above. A labelled memory space is called a variable because it can hold different data at different times in the program.

2

```
10 LET A=3
20 LET A$="SNAILS"
30 LET B=43
40 LET B$="ROBOTS"
```

Don't forget quotation marks.

You use a different kind of label to store letters and symbols in memory spaces. Letters and symbols are called "strings" and you use letters of the alphabet with dollar signs to label them, e.g. C$.*

You put a string in a memory space using LET in the same way as for a number variable, but the letters and symbols must be enclosed in quotation marks, as shown above.

3

```
10 LET B=365
20 LET D$="DAYS IN THE YEAR"
30 LET L$="EXCEPT LEAP YEAR"
40 PRINT B
50 PRINT D$
60 PRINT L$
70 END  ←── Many computers do
             not need an END.
```

To display the information on the screen you use the word PRINT with the name of the variable, e.g. PRINT A$. This short program prints out the information from variables B, D$ and L$.

4

```
RUN
365
DAYS IN THE YEAR
EXCEPT LEAP YEAR
```

You can run the program as many times as you want. Each time the computer will print out the same information. The data in the variables stays the same until you change it.
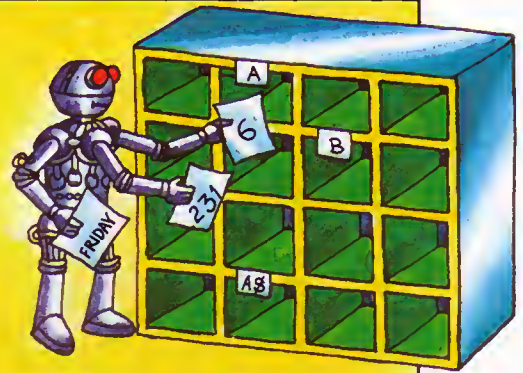
60

*This is pronounced "C dollar" or "C string".

Another way

```
10  READ A
20  READ B
30  READ A$
40  DATA 6, 231, FRIDAY
```

You must have the correct kind of label for numbers and letters.

Commas

Some computers need their data words in quotes.

Another way to store information is with the words READ and DATA, as shown above. The READ lines tell the computer to label memory spaces and the DATA line contains the information.

When you run the program the computer puts each piece of data in a memory space, taking them in order. The items of data must have commas in between so the computer knows how long each one is.*

Some programs

1
```
10  READ Q
20  READ X$
30  DATA 24, CHEESE BURGERS
40  PRINT Q
50  PRINT X$
60  END
RUN
24
CHEESE BURGERS
```

Comma

This is one item of data, including the space.

2
```
10  LET A$="ROBOTS ARE GREAT"
20  LET B$="IF YOU LIKE"
30  LET C$="GREAT METAL IDIOTS"
40  PRINT A$
50  PRINT B$
60  PRINT C$
70  END
RUN
ROBOTS ARE GREAT
IF YOU LIKE
GREAT METAL IDIOTS
```
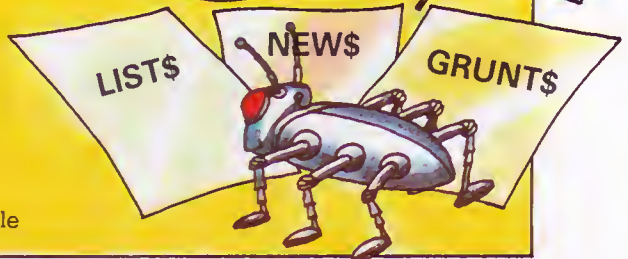
Quotes

Here are two programs, one using READ and DATA and the other using LET to store information in the computer's memory.

More about variables

Number variable

String variable
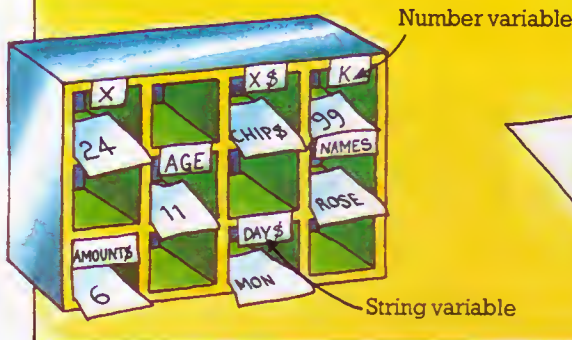
You cannot use these words as variable labels as they contain BASIC words.

LIST$ NEW$ GRUNT$

Variables are labelled spaces in the computer's memory where information is stored. A variable containing numbers is called a number variable and one which contains letters and symbols is called a string variable. The contents of variables can change during the program. Some computers can use words as labels for variables, but not words which contain BASIC words as this would confuse the computer.

61

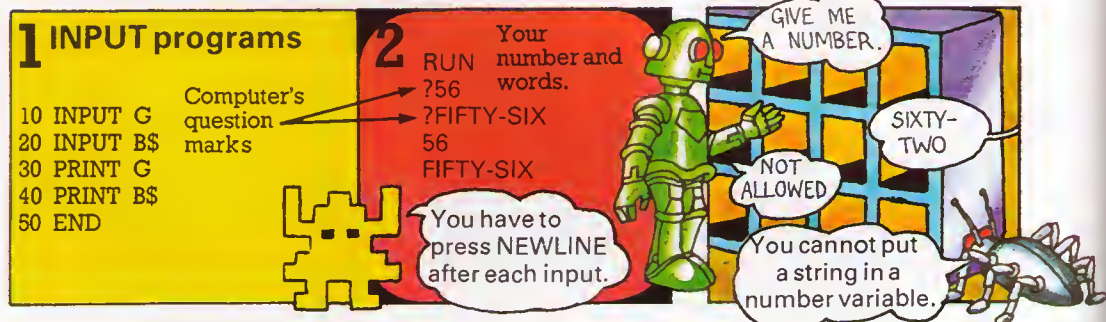*You cannot use this method on the ZX81 computer.

Using INPUT

Another way to give the computer data is with the word
INPUT. This lets you put in information while the program
is running, and you can use different data each time
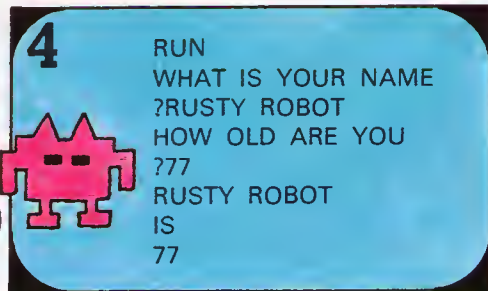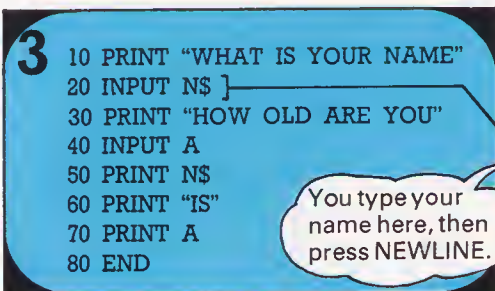you run the program.

```
20 INPUT A
30 INPUT A$
```
For numbers
For words
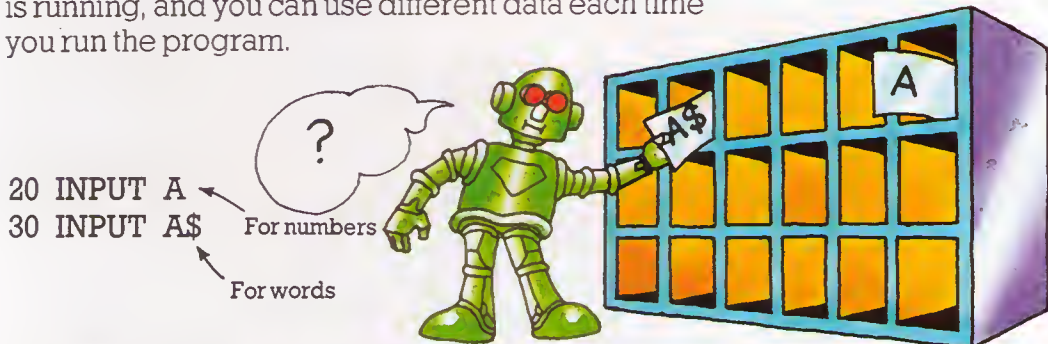
You use INPUT with a label such as A for a
number and A$ for a string. When the
computer meets the word INPUT in a
program it puts the label on a memory
space and asks you for the data, usually by
printing a question mark, or other symbol,
on the screen. Then you type in the data
and the computer stores it in the memory
space and goes on with the rest of the
program.

1 INPUT programs

```
10 INPUT G
20 INPUT B$
30 PRINT G
40 PRINT B$
50 END
```
Computer's
question
marks

2

```
RUN
?56
?FIFTY-SIX
56
FIFTY-SIX
```
Your
number and
words.

GIVE ME
A NUMBER.

SIXTY-
TWO

NOT
ALLOWED

You have to
press NEWLINE
after each input.

You cannot put
a string in a
number variable.

Picture 2 shows what happens when you
run this program. When the computer
meets the word INPUT in line 10 it prints a
question mark on the screen and waits for
you to type in a number for G. Then it
prints another question mark for the
INPUT instruction in line 20. This time you
have to type in words or symbols as the
label B$ told the computer to expect a
string.

3

```
10 PRINT "WHAT IS YOUR NAME"
20 INPUT N$
30 PRINT "HOW OLD ARE YOU"
40 INPUT A
50 PRINT N$
60 PRINT "IS"
70 PRINT A
80 END
```
You type your
name here, then
press NEWLINE.

4

```
RUN
WHAT IS YOUR NAME
?RUSTY ROBOT
HOW OLD ARE YOU
?77
RUSTY ROBOT
IS
77
```

If you have a computer, try typing in this
program, then press RUN to start it off.
When the computer asks you for
information, type in your name and age, or
a silly name and crazy number, as shown
in the sample run above. Try it lots of times
with different data, pressing RUN to start
the program again each time. The
computer always prints exactly what you
put in N$ and A.

Poetry writing program

Now you know enough BASIC to write a poem on a computer. Here is a poetry writing program which uses PRINT and INPUT.

```
10  PRINT "WHAT IS YOUR NAME"
20  INPUT N$
30  PRINT "A POEM BY"
40  PRINT N$
50  PRINT "TYPE IN A WORD"
60  PRINT "THAT RHYMES WITH ME"
70  INPUT A$
80  PRINT "HERE IS THE POEM"
90  PRINT "COMPUTERS USED TO
       FRIGHTEN ME"
100 PRINT "BUT NOW I'M HAPPY AS A"
110 PRINT A$
120 END
```

This line prints out your name. (lines 30-40)

This line prints out your word. (line 110)

```
RUN
WHAT IS YOUR NAME
?SAL
A POEM BY SAL
TYPE IN A WORD
THAT RHYMES WITH ME
?BEE
HERE IS THE POEM
COMPUTERS USED TO
 FRIGHTEN ME
BUT NOW I'M HAPPY AS A
BEE
```

Your name and word.

You type run to try it again with another word.

The program makes the computer ask you your name, then store your reply in N$ and print it out at line 40. It stores the word you choose in A$, then prints it out as part of the poem at line 110. If you have a computer try running the program lots of times, inputting different words at line 70.

Program puzzle

Can you write a program to get the computer to ask you your name and then print hello, followed by your name and a message to you?
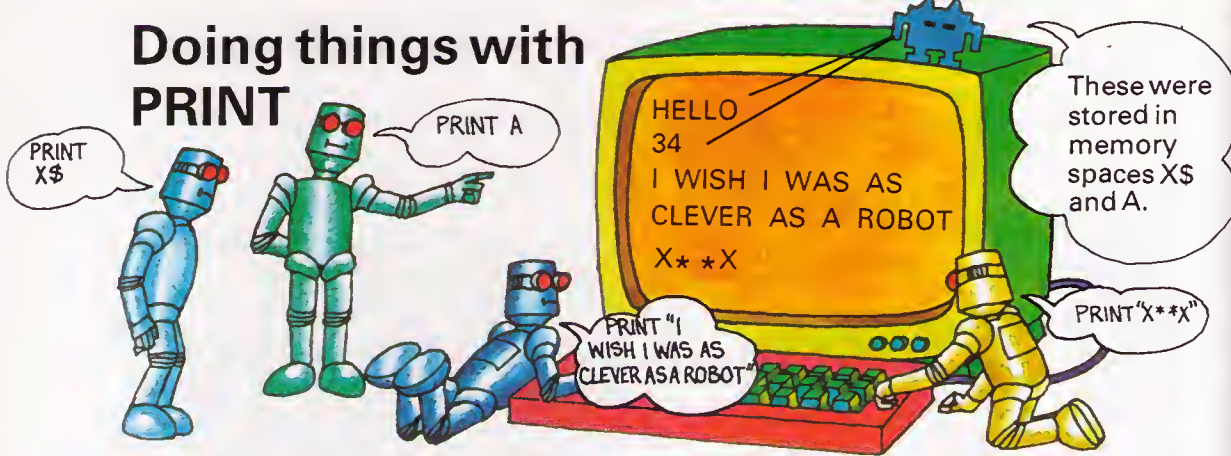
Checklist for typing in programs

1. Before typing in a new program type NEW. This clears any old programs and variables out of the computer's memory.

2. When you are typing in the program, remember to press NEWLINE, or your computer's word, at the end of each line.

3. After typing in the program, check all the lines on the screen to see if there are typing mistakes. Make sure none of the lines are missing, too.

4. Next you can type CLS (or your computer's word) to clear the program off the screen. Then type RUN to start the program.

5. To get the program listing back again to check it or alter a line, type LIST. To display one particular line you can usually type LIST with the line number, but check this command as it varies slightly on different computers.

6. To stop the program while it is running type BREAK or ESCAPE. Check this command in your manual, though, as it varies on different computers. On some computers ESCAPE wipes the whole program out of the computer's memory. To start the program again type RUN.

There are some hints to help you find bugs on pages 90-91.

63

Doing things with PRINT

PRINT X$

PRINT A

These were stored in memory spaces X$ and A.

HELLO
34
I WISH I WAS AS CLEVER AS A ROBOT
X* *X

PRINT "I WISH I WAS AS CLEVER AS A ROBOT"
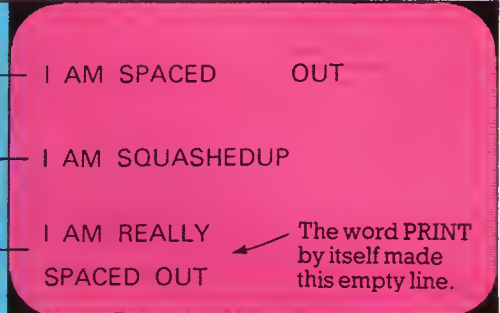
PRINT "X* *X"

So far you have seen how to use PRINT to display words and numbers on the screen, and to print out the contents of variables. Below you can find out how to use commas and semi-colons to space things out on the screen. You can also use

PRINT to do calculations on a computer. You can find out how at the bottom of the page. On the opposite page you can find out more about doing things with variables.
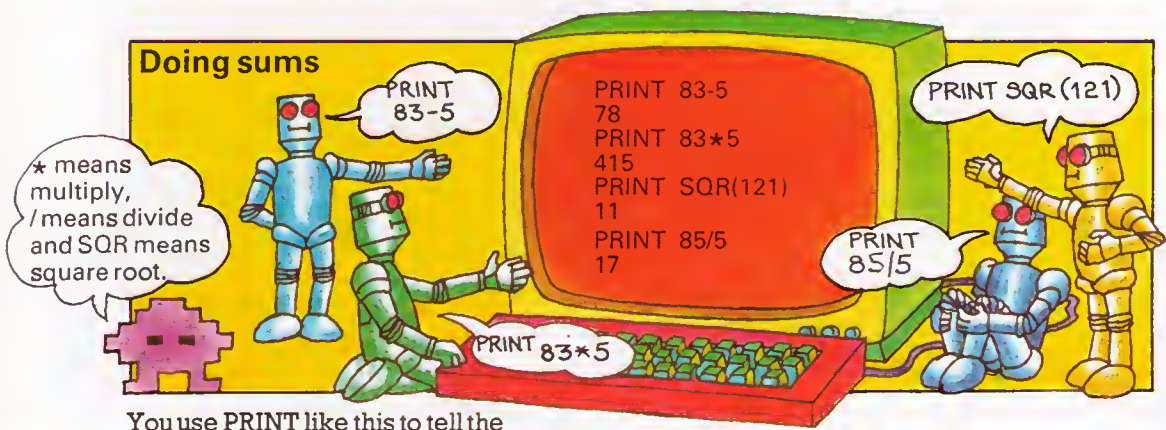
Commas and semi-colons

```
10 PRINT "I AM SPACED",
20 PRINT "OUT"
```
Comma

I AM SPACED   OUT

```
10 PRINT "I AM SQUASHED";
20 PRINT "UP"
```
Semi-colon

I AM SQUASHEDUP

```
10 PRINT "I AM REALLY"
20 PRINT
30 PRINT "SPACED OUT"
```

I AM REALLY

SPACED OUT

The word PRINT by itself made this empty line.

These lines show how you can use commas and semi-colons to tell the computer where to print the next letter. A comma tells it to move along the screen a bit and a semi-colon tells it to stay where it

is. The picture above shows how the lines would be printed on the screen. The word PRINT on a line by itself tells the computer to leave an empty line.

Doing sums

PRINT 83-5

* means multiply, / means divide and SQR means square root.

PRINT SQR (121)

```
PRINT 83-5
78
PRINT 83*5
415
PRINT SQR(121)
11
PRINT 85/5
17
```
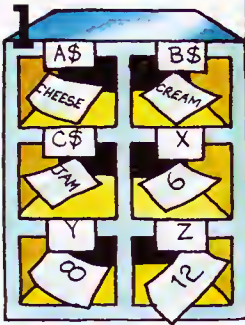
PRINT 85/5

PRINT 83*5

You use PRINT like this to tell the computer to do sums. You use the normal signs for addition and subtraction and * for multiplication and / for division.

The computer can also do more complex mathematical calculations such as sines, cosines, square roots, etc.

More about variables

1

2

Spaces

```
PRINT "I ATE ";X;" PEANUT BUTTER AND ";A$;" SANDWICHES"
I ATE 6 PEANUT BUTTER AND CHEESE SANDWICHES

PRINT "I ATE ";Z;" PEANUT BUTTER AND ";C$;" SANDWICHES"
I ATE 12 PEANUT BUTTER AND JAM SANDWICHES
```

Printing variables by themselves is not very useful. You usually need some words with them to say what they are. To print words and a variable together the words must be in quotation marks as usual, and the variable must have a semi-colon either side of it, as shown above. If you want to space out the information you can use commas instead of semi-colons.

3

```
LET X=X+1

LET C$=C$+"MY"
```

During a program you can change the contents of memory spaces like this. To the computer these statements mean add one to the figure in memory space X and add "MY" to the letters in C$.

4

Spaces

```
PRINT "I ATE ";X;" PEANUT
BUTTER AND ";C$;" SANDWICHES"

I ATE 7 PEANUT BUTTER AND
JAMMY SANDWICHES
```

Next time you ask the computer to print the variables it will display the new words and numbers stored in the memory spaces.

5

```
10 LET A=9
20 LET B=7          Multiply
30 PRINT A*B
40 PRINT A/B
50 END
RUN                 Divide
63
1.28571
```

You can do sums with variables too, as shown in the program above. The computer finds the numbers in the memory spaces, then works out the sums.

Program puzzles

1. Write a program to add numbers to the variables in the program on the left so that it would print out the answers 100 and 1 on one line with a space between.

2. Change lines 30 and 40 so that they print out the numbers, what you are doing to them and the answer, e.g. "7 times 9 is 63".
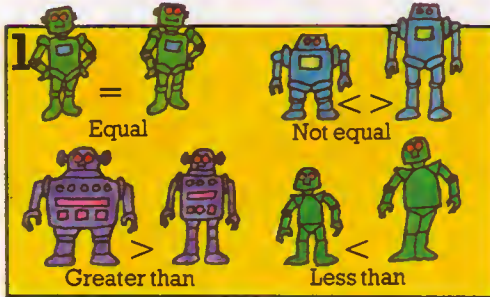
3. Change your answer to the program puzzle on page 63 so it prints your name and the message on one line.

How computers compare things

One of the most useful things a computer can do is to compare pieces of information and then do different things according to the results. To do this you use the words IF... THEN.

1

Equal   Not equal

Greater than   Less than

The computer can do several different tests on information to compare it. The symbols for the tests are shown above. It can test to see if two pieces of data are equal, different, or if one is greater or less than the other.

2

```
IF A=B THEN PRINT "THEY ARE EQUAL"
IF A>B THEN PRINT "A IS BIGGER"
IF A<B THEN PRINT "A IS SMALLER"
IF A<>B THEN PRINT "THEY ARE NOT
                              EQUAL"
```

These lines show how you use the symbols with IF and THEN to make the computer compare two pieces of data. You can compare any kind of data – words, numbers and variables, i.e. the contents of memory spaces, too.

3 Weather program

```
10 PRINT "WHAT'S THE WEATHER LIKE TODAY"
20 INPUT W$
30 IF W$="RAIN" THEN PRINT "UMBRELLA TIME"
40 IF W$="SUNNY" THEN PRINT "GOOD"
50 END
```

If you input these words, nothing will happen.

SNOWY   HOT

4

```
RUN
WHAT'S THE WEATHER
LIKE TODAY
?SUNNY
GOOD
RUN
WHAT'S THE WEATHER
LIKE TODAY
?RAIN
UMBRELLA TIME
```

Here is a program using IF and THEN. At line 20 the computer stores the word you input in variable W$. Then, at lines 30 and 40 it checks to see if the word in W$ is the same as "rain" or "sunny". If it is, it prints

out one of the responses. If you put in a different word at line 20 nothing will happen. You could change the words in lines 30 and 40, though, then try inputting one of the new words.

5 Age program

```
10 PRINT "HOW OLD ARE YOU"
20 INPUT A
30 IF A>16 THEN PRINT "OLD"
40 IF A<16 THEN PRINT "YOUNG"
50 IF A=16 THEN PRINT "JUST RIGHT"
RUN
HOW OLD ARE YOU
?16
JUST RIGHT
```

6 French lesson

```
10 PRINT "HOW DO YOU SAY RED IN
   FRENCH"
20 INPUT A$
30 IF A$="ROUGE" THEN PRINT "CORRECT"
40 IF A$<>"ROUGE" THEN PRINT "NO,
   ROUGE"
RUN
HOW DO YOU SAY RED IN FRENCH
?BLEU
NO, ROUGE
```

In the age program, the computer compares input A with the figure 16. If it is bigger than 16 it prints "old". If it is smaller it prints "young" and if it is 16 it prints "just

right". In the other program the computer prints out one of two different responses depending on whether A$ equals "rouge" or not.

Program puzzle – Can you write a program to get the computer to ask you sums, then either print "correct" or give you the right answer?

Branching programs

1
```
IF A=6 THEN LET A$="SIX"

IF X=Y−2 THEN LET Z=0

IF S=T THEN STOP

IF R<10 THEN GOTO 30
```

This tells the computer to go to line 30.

2
```
10 INPUT K$
20 IF K$="YES" THEN GOTO 100
30 IF K$="NO" THEN GOTO 200
⋮
100 PRINT "YOU TYPED YES"
110 STOP
⋮
200 PRINT "YOU TYPED NO"
210 END
```

These two lines make the computer branch to other parts of the program.

You can give the computer almost any instruction after the word THEN, as shown above. A useful instruction is to make it go to another line. (On most computers, but not the ZX81, you can leave out the word GOTO.) You usually need a STOP instruction in programs with GOTO, or the computer will go on repeating the program endlessly.

Maths program

```
10 PRINT "TYPE IN A NUMBER"
20 INPUT A
30 PRINT "TYPE IN ANOTHER NUMBER"
40 INPUT B
50 PRINT "DO YOU WANT TO"
60 PRINT "ADD, SUBTRACT, MULTIPLY"
65 PRINT "DIVIDE OR STOP"
70 INPUT C$
80 IF C$="ADD" THEN PRINT A+B
90 IF C$="SUBTRACT" THEN PRINT A−B
100 IF C$="MULTIPLY" THEN PRINT A∗B
110 IF C$="DIVIDE" THEN PRINT A/B
120 IF C$="STOP" THEN STOP
130 GOTO 10
```

```
RUN
TYPE IN A NUMBER
?17
TYPE IN ANOTHER NUMBER
?184
DO YOU WANT TO
ADD, SUBTRACT, MULTIPLY
DIVIDE OR STOP
?ADD
201 ◀────── Computer's
TYPE IN A NUMBER    answer
?
```

THE PROGRAM WILL ONLY STOP WHEN YOU INPUT THE WORD STOP
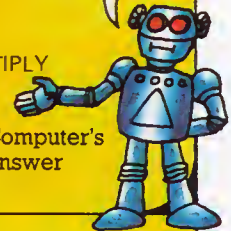
In this program the numbers you type in are stored in A and B and your instructions are stored in C$. At lines 80 to 120 the computer compares C$ with five different words, and when it finds the right word, it carries out the instruction. It passes over all the lines which are not true.

Age guessing program

1
```
10 PRINT "GUESS MY AGE"
20 INPUT G
30 IF G<>14 THEN PRINT "TRY AGAIN"
40 IF G<>14 THEN GOTO 20
50 PRINT "CORRECT"
60 END
```

2
```
RUN
GUESS MY AGE
?15
TRY AGAIN
?14
CORRECT
```

3
```
GUESS MY AGE
?15
YOUNGER THAN THAT
?13
OLDER THAN THAT
?14
CORRECT
```

CAN YOU WRITE THE PROGRAM FOR THIS?

This program will go on repeating itself until G=14. When G=14 the computer will pass over lines 30 and 40 and print "correct". Can you alter the program so that it gives you some clues, as shown in the picture on the right?

Programs with lots of BASIC

The programs on these two pages use most of the BASIC covered so far. The first program is a space game for two people to play with the computer. If you do not have a computer, study the programs and try and follow how they work.

Space commando

```
10  PRINT "ALIEN'S SQUARE ALONG"
20  INPUT A
30  PRINT "ALIEN'S SQUARE UP"
40  INPUT B
50  CLS
60  PRINT "COMMANDO'S SQUARE ALONG"
70  INPUT C
80  PRINT "COMMANDO'S SQUARE UP"
90  INPUT D
100 CLS
110 LET X=SQR((A−C)*(A−C)+(B−D)*(B−D))
120 PRINT "YOU ARE NOW"
130 PRINT X;"SPACE UNITS APART"
140 IF X<1.5 THEN PRINT "ALIEN FOUND"
150 IF X<1.5 THEN STOP
155 PRINT "WHAT ARE YOUR NEW POSITIONS"
160 GOTO 10
170 END
```

Lines 10 to 40 store alien's co-ordinates in A and B.

Line 50 clears alien's numbers off screen.

Lines 60 to 90 store commando's co-ordinates in C and D.

This line works out how far apart they are and stores the answer in X.

If X is less than 1.5 the program stops. If it is more than 1.5 the program repeats.

In this game, one person is a hostile alien and the other is a space commando trying to catch the alien. Each player draws a secret map on which they plot their positions (you can find out how to do this below). They give the computer the grid co-ordinates of their positions and the computer then works out how far apart they are. The players use the computer's figures to help them work out their next moves.

How to play

For their secret map, each player draws a grid of 20 × 20 squares and numbers them as shown on the right. The alien starts in the left side of the grid and the commando starts in the right. Each turn, they can move two squares up, down, sideways or diagonally and then give the computer their new positions. When they are less than 1.5 space units (i.e. squares) apart, the commando has caught the alien.
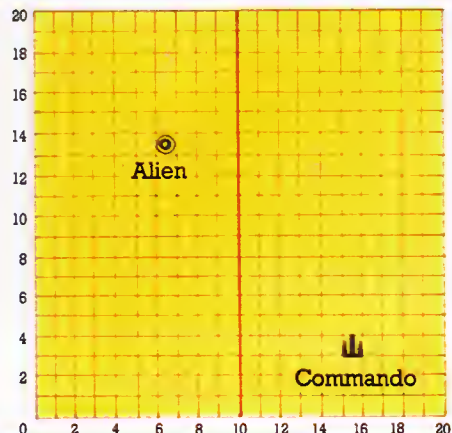
How to make the computer look clever

In this program the computer appears to respond to your answers to its questions. You can see how the program works in the pictures at the bottom of the page. The program uses INPUT in a slightly different way which makes the program shorter and easier to read.

1
```
10 INPUT "GIVE ME A NUMBER"; N
20 INPUT "AND ANOTHER"; M
30 PRINT N; " TIMES "; M;
   " IS "; N*M
```

The BBC micro does not need a semi-colon.

On most computers (not the ZX81) you can make the INPUT line clearer by putting words in quotes before the variable name.

2
```
RUN
GIVE ME A NUMBER?10
AND ANOTHER?8
10 TIMES 8 IS 80
```

On the ZX81 you have to type
```
10 PRINT "GIVE ME A NUMBER"
15 INPUT N
```

When you run the program the input question mark appears after the words.

The program

```
5 LET C=0
10 PRINT "I WOULD LIKE TO TALK TO YOU"
20 INPUT "TELL ME ANYTHING SILLY THAT
   HAPPENED TO YOU THIS WEEK"; A$
30 READ B$
40 PRINT B$;        ← This makes the computer stay on the same line.
50 INPUT C$         Your reply is stored in C$.
60 LET C=C+1
70 IF C=6 THEN GOTO 100
80 GOTO 30
90 DATA WHY, WHY IS THAT
95 DATA WHY, CAN YOU EXPLAIN
98 DATA CAN YOU SAY WHY, WHAT WAS THE REASON
100 PRINT "SO THE REASON YOU TYPED"
110 PRINT "      "; A$
120 PRINT "WAS REALLY GIVEN BY YOUR ANSWER"
130 PRINT "      "; C$
140 PRINT "HOW ODD!"
150 PRINT "RUN ME AGAIN FOR FURTHER
   ENLIGHTENMENT"
160 END
```

This is the new input way. Your reply is stored in A$.

At line 30 the computer looks for the first line with DATA and takes the first item and puts it in B$.

Variable C in lines 60 and 70 is a counter. It keeps count of the number of times the program is repeated. When C = 6 all the data items have been used and the computer moves on to line 100.

Line 80 makes the computer go back to line 30 and replace the data in B$ with the next item in the data list.

The spaces in lines 110 and 130 leave spaces on the screen before your replies. It does not matter how many spaces you leave in the program.

How it works

69

Drawing pictures

A computer makes pictures by lighting up little rectangles on the screen. Each rectangle is called a pixel and each pixel needs a separate instruction from the computer to switch it on. Most computers can also make the pixels different colours.

On these two pages you can find out how to use BASIC to make simple pictures on the screen. The instructions given here are for single colour pictures only.
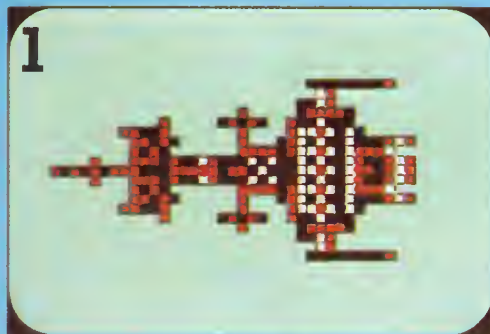

You can usually see the pixels in a computer picture. A computer with a large memory, though, can make pictures with thousands of very small pixels. These pictures are called high resolution graphics.
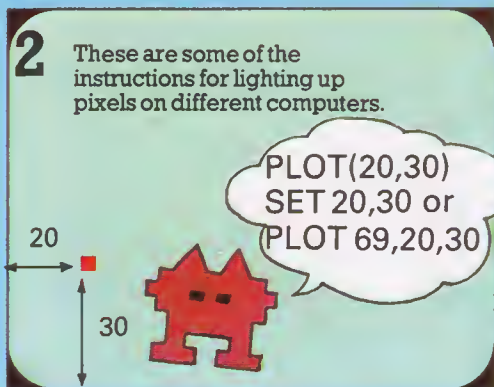
These are some of the instructions for lighting up pixels on different computers.

PLOT(20,30)
SET 20,30 or
PLOT 69,20,30

The instruction for lighting up a pixel varies on different computers, but it is usually something like PLOT (X, Y). X and Y are the pixel's co-ordinates and X is the number of pixels along and Y is the number of pixels up.

On some computers numbering starts here.
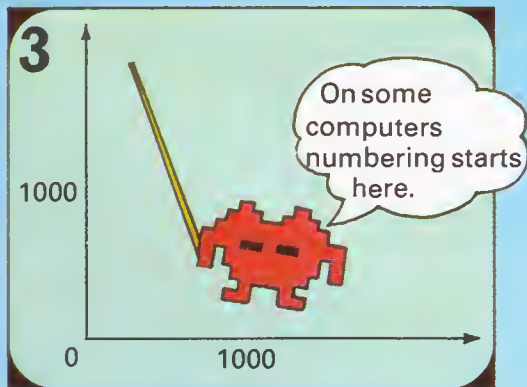
On a computer with high resolution graphics you may be able to plot 1000 points along the screen and 1000 up. A less powerful computer has about 60×40. (If you have a computer, check the size of your screen as you may get a bug if you plot outside its range.)

Remember – some computers need a general graphics instruction.
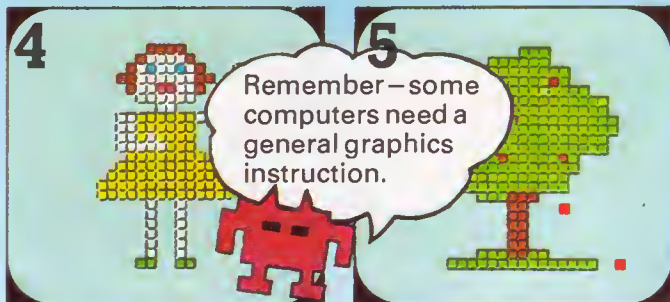
Pictures made by a computer are usually called graphics. Some computers need a special command before you do graphics. For instance, on the BBC micro you need the word MODE with a number.*

UNPLOT (20,30)
RESET 20,30

You can also switch a pixel off with a command such as UNPLOT (X, Y). In the programs in this book we use PLOT and UNPLOT. If you have a computer check these commands in your manual.

70

*For the programs in this book use MODE 5 on the BBC micro with the plot command PLOT 69, X, Y. For unplotting use PLOT 71, X, Y.

Plotting program

PLOT command varies on different computers.

1
```
10 PRINT "TYPE IN
   TWO NUMBERS"
20 INPUT X
30 INPUT Y
40 PLOT (X,Y)
50 GOTO 10
```

You have to press NEWLINE or RETURN after inputting each number

2
```
RUN
TYPE IN TWO NUMBERS
?24
?24          ← 1st pixel
TYPE IN TWO NUMBERS
?30
?15          ← 2nd pixel
```

This short program asks you for two numbers, then plots the pixel with those numbers as co-ordinates. If you try this program make sure the numbers you type in are within the range of your computer.

Line 50 makes the program repeat itself endlessly and the only way to stop it is with BREAK (or the computer's own word). Can you insert a counter (see page 69) to make it run, say, six times.

Plotting a picture

```
10 LET X=10
20 LET Y=10
30 PLOT (X,Y)        This plots
40 LET X=X+1         a diagonal
50 LET Y=Y-1         line going down.
60 IF X<14 THEN GOTO 30

100 LET Y=Y+1        This plots
110 LET X=X+1        a diagonal
120 PLOT (X,Y)       line going up.
130 IF X<20 THEN GOTO 100
```

Adding 1 to X and not to Y plots a horizontal line

Adding 1 to Y and not to X plots a vertical line.

First you need to draw the picture on squared paper and work out the co-ordinates of the squares.

Then you can work out the program to plot all the squares. By giving X and Y starting values, then adding to them or subtracting from them, and repeating parts of the program, you can make the computer plot sequences of pixels as shown above.

X−5, Y+10 moves picture up and left.

X*3, Y*3 makes an exploded image.

Original picture
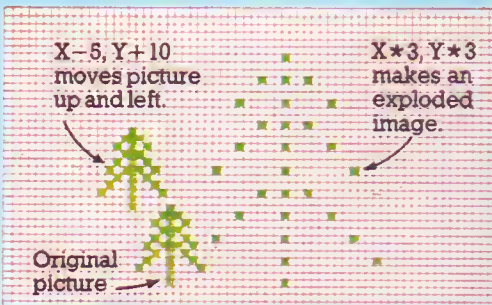
After writing the program it is easy to change the picture by altering the numbers. You can move it to a different place on the screen by changing the starting values, or multiply all the numbers by three to make an "exploded image".

Another way
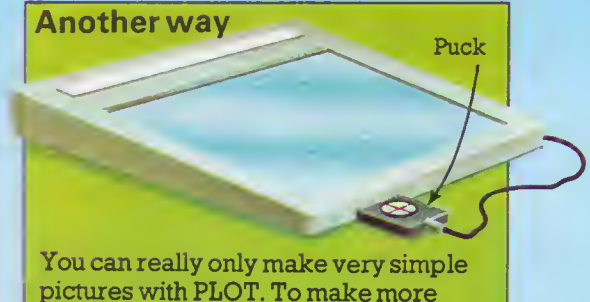
Puck

You can really only make very simple pictures with PLOT. To make more complicated ones you need special equipment such as a graphics tablet. You place a drawing on the tablet and trace over it with a special device called a "puck". This automatically reads the co-ordinates into the computer.

71

★ **Program puzzle** – Can you write a program to plot your initial on the screen? There is a sample program on page 92.
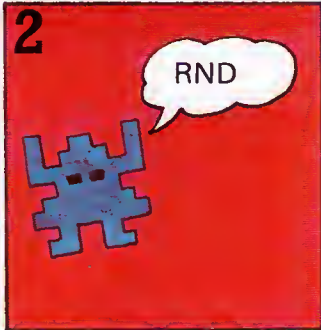
Playing games

1

When you throw a pair of dice you cannot predict what the numbers will be. The chances are equal that the numbers will be anything from one to six. You can produce unpredictable numbers on a computer. They are called random numbers.

The computer contains a special program for producing random numbers. Sometimes it repeats the same number several times, but in sequences of lots of random numbers, the number of times each number is picked is about even.

2

RND

3

RND(0)

RND(1)

4

PRINT RND
.662741814
PRINT RND(99)
77

Or RND(0), or RND(1) on some computers.

To make the computer produce a random number you use the word RND. Some computers need a 1 or 0 in brackets after the word. If you have a computer, check your manual for the correct command.
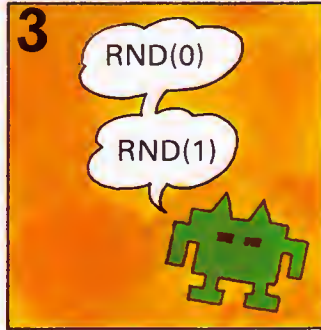
The RND instruction always produces a number below one. On some computers you can put a number in brackets after RND, e.g. RND(99). This makes it produce a whole number between 1 and the number in brackets.

5

INT (RND(1) * 99 + 1)

77

INT (RND(0) * 99 + 1)

45
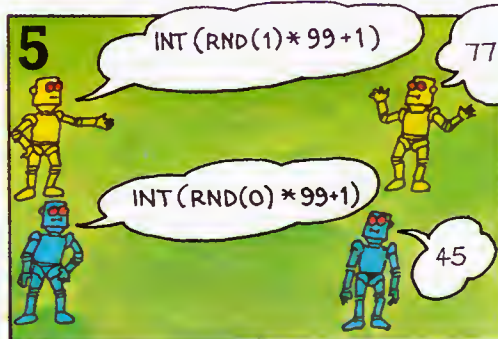
On other computers you need the word INT (short for integer, meaning whole number) followed by the RND instruction (either RND(1) or RND(0) on different computers). Then you multiply by the highest number you want and add one so the number is above one.

6

LET R = INT (RND(1) * 60 + 1)

37

R
37
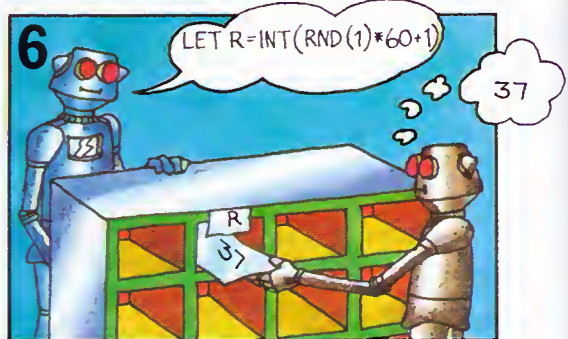
This instruction means pick a random number and store it in variable R. In the programs in this book we use INT(RND(1) * 60 + 1) to mean pick a random number between 1 and 60. You may need to convert this instruction for some computers.

72

Program puzzle – Can you work out how to make the computer pick a random number between 10 and 20?

Space attack

This is a program for a game using random numbers. In the game you are on a star ship being attacked by a wave of alien fighters. Your ship's computer locates the aliens and gives you their coded positions. To hit each alien you have to work out the firing range by multiplying the codes and typing in the answer.

```
10  LET  C=0
20  LET  A=INT(RND(1)*20+1)
30  LET  B=INT(RND(1)*20+1)
40  PRINT  "ALIEN SHIP'S CODES"
45  PRINT  "ARE ";A,B;" FIRE"
50  INPUT  X
60  LET  C=C+1
70  IF  X=A*B  THEN  PRINT "ALIEN SHIP
       DESTROYED"
80  IF  X<>A*B  THEN  PRINT "MISSED"
90  IF  C<6  THEN  GOTO  20
100 END
```

C is a counter to count the number of times the program is repeated. Each time, line 60 adds 1 to C.

These two lines produce random numbers for the alien ship's codes and store them in A and B.

Your number is stored in X.

In lines 70 and 80 the computer checks to see if you got the right answer.

This line repeats the program if C is less than 6.

Running the program

The picture on the right shows what happens when you run the program. If you type in the correct answer for the two numbers multiplied together the computer will print "alien ship destroyed". If your answer does not equal A×B the computer prints "missed".

```
RUN
ALIEN  SHIP'S  CODES
ARE  17      3  FIRE
?41
MISSED
ALIEN  SHIP'S  CODES
ARE  11      5  FIRE
?55
ALIEN  SHIP  DESTROYED
ALIEN  SHIP'S  CODES
ARE  13      6  FIRE
```

The comma in line 45 spaced out the numbers like this.

★ Program puzzle

Can you add another counter to the program to count your number of hits and print out your score at the end of the game? You need to set up a variable called S and give it a value of 0 to start with, then add 1 for each hit.

Random pattern program

```
5   CLS
10  LET  X=INT(RND(1)*30+1)
20  LET  Y=INT(RND(1)*30+1)
30  PLOT  (X,Y)
40  GOTO  10
```

This clears the program off the screen before the pixels are plotted.

The random numbers must fit on the computer's screen.

This line makes the program repeat endlessly.

Computers' commands for CLS, RND and PLOT may vary and some will need a graphics line.

This program uses random numbers to plot spots of light on the screen. Lines 10 and 20 produce random numbers between 1 and 30 and store them in X and Y. Line 30 then plots the pixel with co-ordinates X,Y. As the screen fills up you see less pixels appearing as many of them are already plotted. To stop the program you have to type BREAK or ESCAPE, or another word on different computers.

Making loops

You often need the computer to do the same thing several times in a program. On page 69 you can see how to make it repeat part of a program using GOTO and a variable which acts as a counter. Another way is to repeat the same lines several times using the words FOR . . . TO and NEXT. This is called making a loop.

1 Hello loop

```
10  FOR J=1 TO 6
20  PRINT "HELLO"     Loop
30  NEXT J
40  END
```

This program has a loop from lines 10 to 30 which makes the computer repeat line 20 six times. The letter J is a variable and line 10 tells the computer to set J at 1 on the first run through the program, 2 the next time, then 3, etc., up to 6. Line 20 tells it to print the word
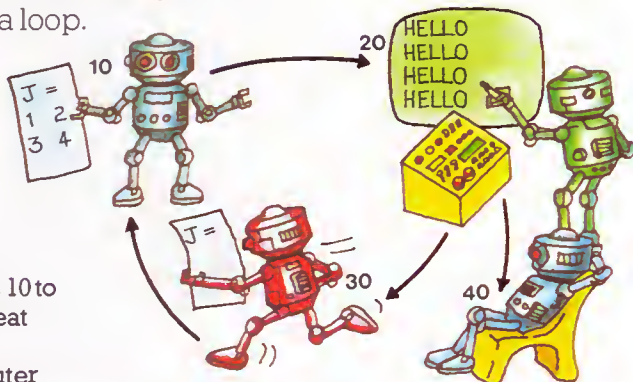
hello and line 30 tells it to go back and find the next value for J. When J=6 the computer goes on to line 40.

2 Silly sums program

```
10  FOR J=1 TO 8          Loop
20  PRINT "2 PLUS 2 IS 5"
30  NEXT J
40  PRINT
50  PRINT "ONLY JOKING!"
60  END
```

Some computers do not have an exclamation mark and you can leave it out.

```
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5
2 PLUS 2 IS 5

ONLY JOKING!
```

In this program, the loop from lines 10 to 30 makes the computer repeat line 20 eight times. Each time it passes through line 20 it prints out the same silly

sum. After doing it eight times the computer carries on with the rest of the program. Line 40 just makes it leave an empty line.

## 3 Eight times table program

```
10  PRINT "THE EIGHT TIMES TABLE"
20  FOR J=1 TO 12
30  PRINT J*8          Loop
40  NEXT J
50  END
```

This time J is used to count the number of loops and also as part of the sum J*8. Line 20 tells the computer to set J at 1, then 2, 3, etc, up to 12. Line 30 takes the current value of J, multiplies it by 8 and prints out the answer. Then line 40 sends the computer back to line 20 to find the next value of J.
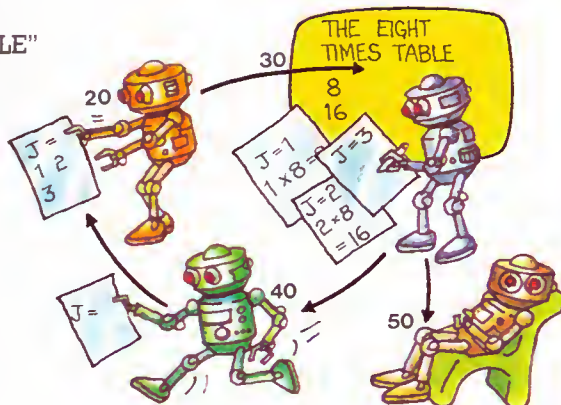
Making patterns
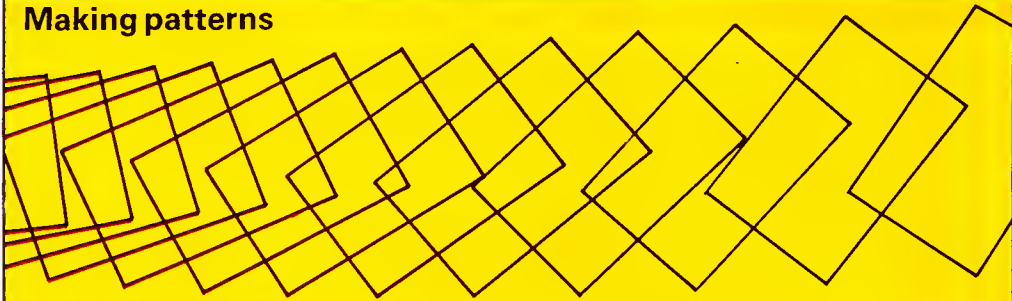
FOR . . . NEXT loops are useful for making patterns, like this, of a simple shape repeated lots of times. The program for this pattern is too long to write out here in full, but it would look something like this:

```
10  FOR I=1 TO 45
20  Draw a rectangle and change its
    position a little each time.
30  NEXT I
40  END
```

Steps

Sometimes it is useful to change the value of J by amounts other than 1. For instance, you may want to go up in 3s or down in 7s. To do this you use the word STEP. In the following program STEP − 1 makes J go down by 1 each time the computer passes through the loop in lines 10 to 40.

Greedy computer program

The figure 2 stops the loop after J=2 (i.e. when there is one pie left).

```
5   CLS
10  FOR J=7 TO 2 STEP −1
20  PRINT "THERE ARE ";J;" PIES LEFT"
30  NEXT J                      Loop
40  PRINT
50  PRINT "I SHALL EXPLODE"
60  FOR K=1 TO 1000
70  REM: DO NOTHING
80  NEXT K                      Loop
90  PRINT
100 PRINT "B A N G S P L A T T"
```

```
THERE ARE 7 PIES LEFT
THERE ARE 6 PIES LEFT
THERE ARE 5 PIES LEFT
THERE ARE 4 PIES LEFT
THERE ARE 3 PIES LEFT
THERE ARE 2 PIES LEFT

I SHALL EXPLODE

B A N G S P L A T T
```

Some computers are slower than others and they need a lower figure such as 500 or 250 in line 60.

There are two loops in this program. The one from lines 10 to 30 makes the computer print line 20 six times. Each time, the value of J is reduced by one and the figure for J is printed in line 20. In the loop from lines 60 to 80 the computer does not have to do anything. It just runs through all the values for K from 1 to 1000 and this makes it pause for a moment. Lines which start with REM (short for remark) are ignored by the computer and are useful to remind you what the program is doing.

Program puzzles

1. Can you alter the eight times table program on the left to make it display "1×8=" as well as the answer?

2. Can you write a program for the "N" times table, that is, a program which works out the tables for any number you type into the computer? First you need to get the computer to ask you for a number, N. Then use a loop to work out and display the tables. If you want, include some lines at the end of the program so it asks you if you want the tables for another number and the program repeats itself.
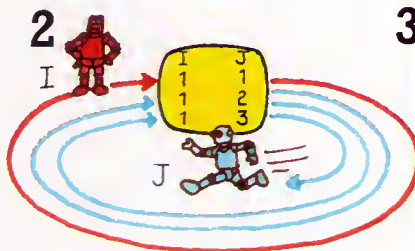
75

Tricks with loops

Here are some more programs using loops. Below you can find out how you can use loops within loops to repeat several things at the same time. These are called nested loops.

1 Nested loops

```
 5 PRINT "I", "J"
10 FOR I=1 TO 3
20 FOR J=1 TO 3
30 PRINT I, J    J loop
40 NEXT J        I loop
50 NEXT I
60 END
```

2

I

J

3

I	J
1	1
1	2
1	3
2	1
2	2
2	3
3	1
3	2
3	3

This program has an I loop and a J loop. The J loop is nested inside the I loop and for each time that the I loop is carried out, the J loop is repeated three times, printing out the value for J each time. The picture above shows the result of this program. The commas spaced the figures out like this.

Computer clock

```
 5 CLS
10 LET M=0
20 LET S=0
30 FOR M=0 TO 59
40 FOR S=0 TO 59
50 PRINT M;":";S
60 CLS           Seconds
70 NEXT S        loop
80 NEXT M        Minutes
90 END           loop
```

0:45

To set the time use this delay loop:
54 FOR Z=1 TO 100
58 NEXT Z

Inside a computer there is an electronic "clock" which sets the rhythm for all the computer's work. The clock pulses at between one and four million pulses a second. This program makes the computer behave like a digital clock. It has nested loops, one to count the seconds and one to count the minutes.

Bugs in loops

```
10 FOR I=1 TO 4
20 FOR J=1 TO 4
30 PRINT I
40 PRINT J
50 NEXT I
60 NEXT J
```

Both parts of a nested loop must be inside the other one.

The seconds loop is carried out 59 times for each minute loop. If you try this program on a computer it might run very fast at first. You need to put in an extra "delay loop", then set it by changing the figure in the loop so your computer clock "ticks" at the same rate as a real one.

Random number tester

```
 10 FOR I=1 TO 1000
 20 LET R=INT(RND(1)*6+1)
 30 IF R=1 THEN LET A=A+1
 40 IF R=2 THEN LET B=B+1
 50 IF R=3 THEN LET C=C+1
 60 IF R=4 THEN LET D=D+1
 70 IF R=5 THEN LET E=E+1
 80 IF R=6 THEN LET F=F+1
 90 NEXT 1
100 PRINT "FINISHED"
110 PRINT A, B, C
120 PRINT D, E, F
130 END
```

This program takes a long, long time. You can make it shorter by changing the number in line 10 to 500 or even 250.

```
RUN
FINISHED
162        168        167
160        187        156
```

This program shows if RND really works. The loop from lines 10 to 90 makes the computer pick a random number between 1 and 6 a thousand times. It keeps count of how often each number is picked in the variables A to F, then prints out the results.*

76

*On some computers, e.g. ZX81 and BBC micro, you need some extra lines at the beginning of the program to set each variable to 0.

Pattern repeat program

This program uses nested loops to repeat a small pattern all over the screen. The program looks quite complicated but if you read it through carefully and work out what each line does, you will soon see how it works. The shape of the pattern is decided by random numbers and will be different each time you run the program.

For computers which have high resolution graphics use larger random numbers, e.g. on BBC micro change figure in lines 10 to 40, to 60.

```
5  CLS
10 LET A=INT(RND(1)*6+1)
20 LET B=INT(RND(1)*7+1)
30 LET C=INT(RND(1)*6+1)
40 LET D=INT(RND(1)*4+1)
50 INPUT "HOW MANY POINTS
   ACROSS THE SCREEN";W
60 INPUT "HOW MANY UP";V
65 CLS
70 FOR I=0 TO V STEP V/6
80 FOR J=0 TO W STEP W/6
90 PLOT (J+A,I+B)
100 PLOT (J+A,I+C)
110 PLOT (J+C,I+D)
120 PLOT (J+B,I+D)
130 NEXT J
140 NEXT I
150 END
```

J loop

I loop

These lines choose the random numbers for the pattern and store them in A, B, C and D.

Lines 50 and 60 ask for the width (W) and height (V) of your screen.

The I loop counts the number of times the pattern is repeated up the screen. Each time, I is increased by the height of the screen (V) divided by 6, so the pattern is repeated six times up the screen.

Each time the loops are repeated, lines 90 to 120 tell the computer to plot four pixels using the current values for I and J plus the random numbers.

The J loop counts the number of times the pattern is repeated across the screen. It works in the same way as the I loop.

1 How it works
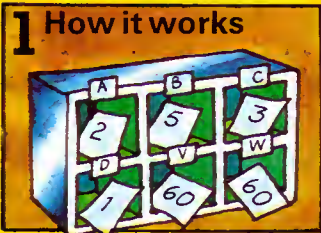
Imagine that the computer has chosen the random numbers 2, 5, 3 and 1 and that the width and height of the screen are both 60.

2

First pattern — Second pattern
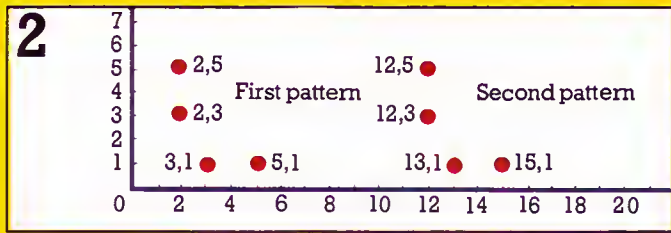
On the first run through the program I and J are 0 so the computer plots the first pattern of dots using only the random numbers. Line 130 sends it back to find the next value for J which is J+60/6, i.e. 10. Then it plots the second pattern using the random numbers plus 10 for J. This repeats the pattern along the screen.

3
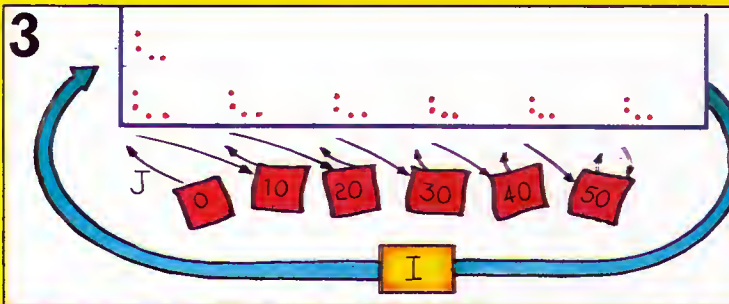
If you try this program and it does not work, try it again with smaller numbers for V and W.

The computer repeats the J loop six times, each time adding 10 to J and so plotting the pattern further along the screen. It then goes ba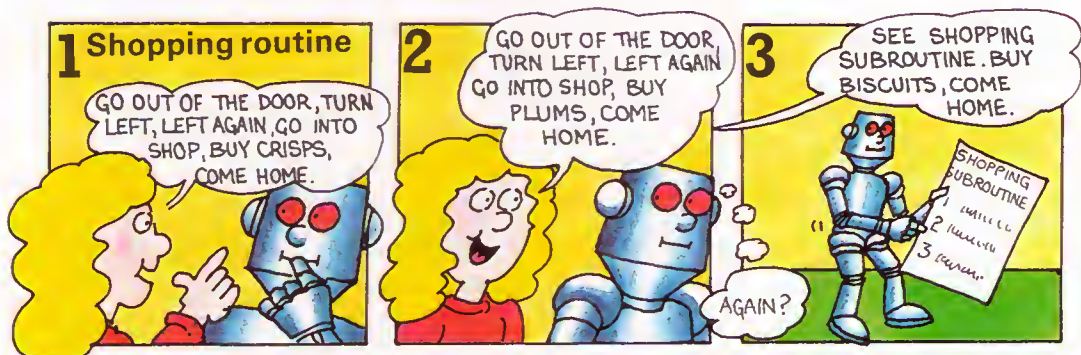ck to find the next value for I which is 10. J is set to 0 again and the computer plots the next line of patterns using 10 for I and increasing J by 10 each time as before.

Program puzzle Can you write a pattern repeat program which repeats a space invader shape over the screen? There are some hints to help you on page 93.
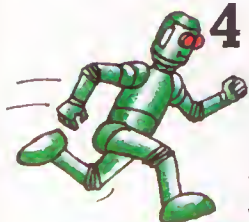
Subroutines

A subroutine is a sort of mini-program within a program. It carries out a particular task, such as adding numbers or keeping a score, and you can send the computer to it whenever you want this task carried out. This saves writing out the program lines each time and makes the program shorter and easier to read and type into the computer.

1 Shopping routine

GO OUT OF THE DOOR, TURN LEFT, LEFT AGAIN, GO INTO SHOP, BUY CRISPS, COME HOME.

2

GO OUT OF THE DOOR, TURN LEFT, LEFT AGAIN GO INTO SHOP, BUY PLUMS, COME HOME.

3

SEE SHOPPING SUBROUTINE. BUY BISCUITS, COME HOME.

AGAIN?

SHOPPING SUBROUTINE

Suppose you had a robot helper whom you could program to run errands for you. If you wanted something from the shop you would have to give it precise instructions telling it how to get there.

Each time you wanted the robot to buy something you would have to give it the same instructions. It would be much simpler to give the robot a shopping subroutine and tell it to refer to it each time.

4 Shopping program

```
10  PRINT "WHAT DO YOU WANT FROM THE SHOP"
20  INPUT X$
30  GOSUB 100
40  PRINT "ANYTHING ELSE"
50  INPUT M$
60  IF M$="YES" THEN GOTO 10
70  STOP

100  REM: SHOP SUBROUTINE
110  PRINT "GO OUT, TURN LEFT"
120  PRINT "LEFT AGAIN, ENTER SHOP"
130  PRINT "BUY ";X$;" COME HOME"
140  RETURN
```

Line 30 sends the computer to the first line of the subroutine.

You need the word STOP at the end of the main program to stop the computer carrying on into the subroutine.

It is useful to label a subroutine with a REM line so you know what it is for.

This sends the computer back to line 40 – the line after GOSUB.

If you forget the RETURN line you get a bug.

In BASIC, to tell the computer to go to a subroutine you use the word GOSUB with the word RETURN at the end of the subroutine. GOSUB should be followed by the number of the first line of the subro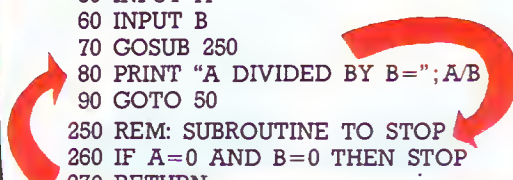utine. RETURN does not need a line number. The computer automatically goes back to the instruction after the one where it left the main part of the program. You can send the computer to a subroutine anywhere in the program as many times as you like.

78

Gosub programs

A subroutine is useful for carrying out any task which you want to repeat several times at different stages in the program. Here are some more programs with subroutines.
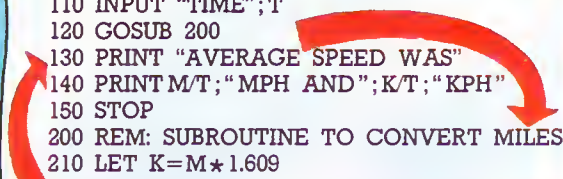
Numbers program

```
50 INPUT A
60 INPUT B
70 GOSUB 250
80 PRINT "A DIVIDED BY B=";A/B
90 GOTO 50
250 REM: SUBROUTINE TO STOP
260 IF A=0 AND B=0 THEN STOP
270 RETURN
```

This subroutine provides an escape from the program. If you want to stop dividing you input 0 at lines 50 and 60. This program does not need STOP before the subroutine as line 90 sends it back.

Conversion program
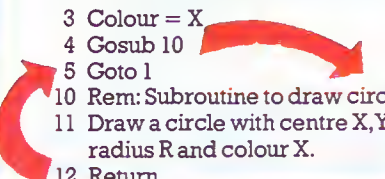
```
100 INPUT "DISTANCE";M
110 INPUT "TIME";T
120 GOSUB 200
130 PRINT "AVERAGE SPEED WAS"
140 PRINT M/T;"MPH AND";K/T;"KPH"
150 STOP
200 REM: SUBROUTINE TO CONVERT MILES
210 LET K=M*1.609
220 RETURN
```

This is a subroutine to convert miles to kilometres. You can often use the same subroutine in lots of different programs. Check that you use the same variable names, though.

Circles program

```
1  Centre of circle = X,Y
2  Radius of circle = R
3  Colour = X
4  Gosub 10
5  Goto 1
10 Rem: Subroutine to draw circles
11 Draw a circle with centre X,Y;
   radius R and colour X.
12 Return
```

This program is in English, not BASIC, to give you the general idea.
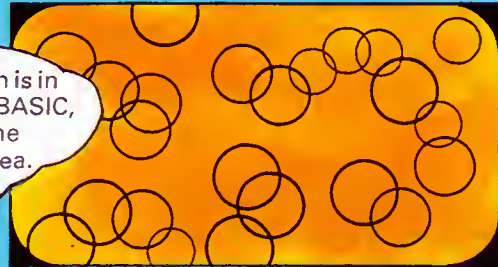
Subroutines are useful in graphics programs like this to draw diagrams with numbers worked out in the main part of the program. With this program you could draw lots of different circles by giving the computer different information in lines 1 to 5.

Quiz program

```
5  LET C=0
10 PRINT "WHEN WERE THESE THINGS INVENTED?"
20 READ C$, F
30 PRINT C$
40 INPUT A
50 LET C=C+1
60 IF C=3 THEN STOP
70 GOSUB 100
80 GOTO 10

100 REM: ANSWERS SUBROUTINE
120 IF ABS(A−F)<10 THEN PRINT "OK"
110 IF ABS(A−F)>10 THEN PRINT "NO"
130 PRINT "TRY ANOTHER ONE"
140 RETURN

200 DATA TELEPHONE, 1876, PRINTING PRESS, 1450, BICYCLE, 1791
```

At line 20 the computer looks for the data line and puts the first word item in C$ and the first number item in F.

This prints out the word in C$.

The counter C stops the program after it has repeated three times as there are only three data items for C$ and F.

This is the subroutine.

Each time the program is repeated the words and numbers in C$ and F are replaced by the next pair of data items.

This program uses a subroutine to check the answers to questions. The correct answers are stored in F and the person's answers go in A. In lines 100 and 110 of the subroutine the computer compares A with F. The word ABS stands for "absolute" and it makes the computer check the difference between the numbers in A and F (it ignores any minus signs). If the difference is less than 10 it prints "OK". If it is more than 10 it prints "NO".

79

Doing things with words

Most computers can examine the words stored in variables and do various things with them. They can check the contents of a variable and see if it contains a particular word or letter. This is useful for checking the words input by someone using the program. Computers can also rearrange the letters or words in a different order and add them to letters in other variables. Below you can find out how you do these things in BASIC.

1

On most computers, but not the ZX81, you can leave out the word LET.

```
10  A$="I AM STUPID"
20  B$="ONLY FOOLS THINK"
30  C$=B$+" "+A$
RUN
ONLY FOOLS THINK I AM STUPID
```

2

```
PRINT LEFT$(B$,4)
ONLY
PRINT LEFT$(B$,4)+" "+A$

ONLY I AM STUPID
```

You can add the contents of two variables like this. You need the space between quotation marks to leave a space between the words.

You can also add parts of variables, like this. LEFT$(B$,4) means take the first four letters from the left of B$.

3

```
PRINT RIGHT$(A$,6)
STUPID
```

4

```
PRINT MID$(B$,6,5)
FOOLS
```

To tell the computer to use letters from the right you use RIGHT$ with the name of the string and the numbers of letters you want.

When you are counting letters, count spaces and punctuation too.

This tells the computer to take the middle letters. The first number tells it where to start and the second tells it how many letters to take.

5

```
10  K$="DING DONG!"
20  PRINT LEN(K$)
RUN
10
```

You can also find out the length of a string – the number of letters, spaces and symbols it contains. To do this you use LEN, short for length.

Note for Sinclair users

```
PRINT A$(6 TO 11)
STUPID
PRINT B$(14 TO 16)
INK
```

This means take letters numbers 6 to 11.

The Sinclair computers do not use LEFT$, RIGHT$ and MID$, but you can tell the computer to take any letters you want as shown above.

IF A$="COMPUTER BOOK" what is LEFT$ (A$,8)? RIGHT$ (A$,10)? MID$ (A$,5,8)?

80

Codemaker program

This program automatically puts words into code. Similar, but much more complex programs are used by intelligence services to write and crack codes.
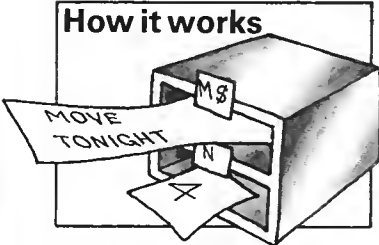
The easiest way to understand this program is to write a secret message on a piece of paper, then work through the lines of the program carrying out the computer's tasks on your message and writing them down.

```
5 LET C$="  "
7 LET D$="  "
10 PRINT "TYPE IN A SHORT MESSAGE"
20 INPUT M$
30 PRINT "NOW TYPE IN A SECRET
   NUMBER BETWEEN 2 AND ";LEN(M$)-1
40 INPUT N
50 LET A$=RIGHT$(M$,N)
60 LET B$=LEFT$(M$,LEN(M$)-N)
70 LET M$=A$+B$
80 FOR I=1 TO LEN(M$) STEP 2
90 LET C$=C$+MID$(M$,I,1)
100 NEXT I
110 FOR J=2 TO LEN(M$) STEP 2
120 LET D$=D$+MID$(M$,J,1)
130 NEXT J
140 LET M$=C$+D$
150 PRINT "CODED MESSAGE IS"
160 PRINT M$
170 END
```

Sets up empty string variables.

This means the length of your message minus 1.

N (your secret number) letters from the right of M$.

The length of M$ minus N number of letters from the left of M$ (i.e. the rest of the letters).

Replaces the letters in M$ with A$+B$.

From 1 to the number of letters in your message going up in twos, i.e. 1, 3, 5, etc. Each time the I loop is repeated line 90 takes one letter from position I of M$ and puts it in C$.

From 2 to the number of letters in your message, going up in twos, i.e. 2, 4, 6, etc. Works in the same way as the I loop.
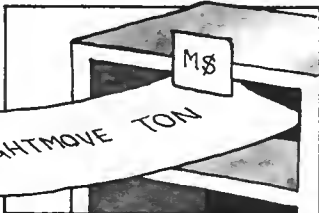
Replaces the letters in M$ again.

How it works

Suppose your message is "Move tonight" and your secret number is 4. These are stored in M$ and N.
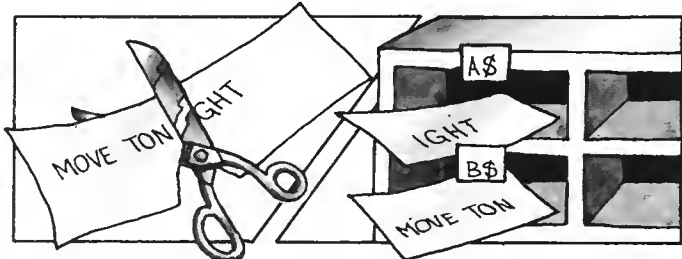
In lines 50 and 60 the computer uses your secret number to divide the message. At line 50 it takes four letters from the right of the message and puts them in A$. At line 60 it puts the rest of the letters in B$.


At line 70 it adds A$ and B$. This puts the letters from the end of the message at the front.
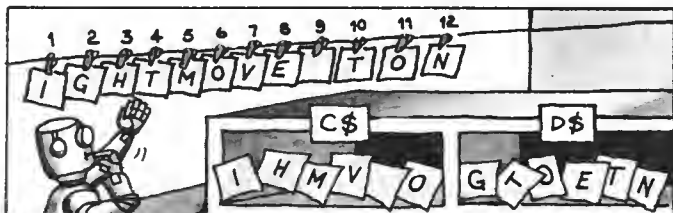
Each time the I loop repeats it puts an odd-numbered letter in C$ (e.g. I, H, M, etc.). Each time the J loop repeats it puts an even numbered letter in D$ (e.g. G, T, O, etc.). Then it adds C$ and D$ to make the coded message.
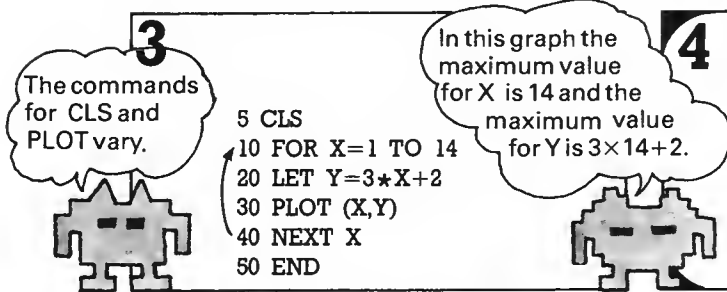
81

Graphs and symbols

You can program a computer to present information in all kinds of different ways, for instance, as words, numbers, pictures or graphs. Complicated information can be made much easier to understand if you illustrate it with graphs, pictures and symbols.


1 Drawing a graph
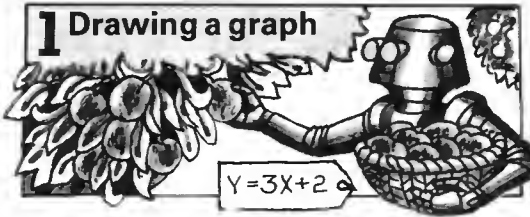
Y=3X+2

Imagine a peach tree whose yield of fruit increases each year in relation to its age. This can be expressed as an equation, say $Y=3X+2$ (Y is the yield and X is the age). It is hard to grasp what this means, though, and drawing a graph would help.

2
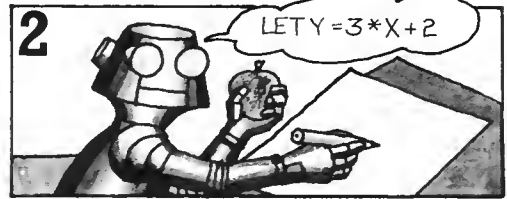
LET Y = 3 * X + 2

With a computer it is very easy to draw a graph of the way Y changes in relation to X. To plot the graph you need to find the value of Y for each value of X. You can do this very easily in a program using the statement LET $Y=3*X+2$.

3

The commands for CLS and PLOT vary.

In this graph the maximum value for X is 14 and the maximum value for Y is $3\times14+2$.

```
 5 CLS
10 FOR X=1 TO 14
20 LET Y=3*X+2
30 PLOT (X,Y)
40 NEXT X
50 END
```

This is the program for drawing this graph. The loop sets X at all the values from 1 to 14. Each time the loop is repeated, line 20 uses the value of X to

4

This is the graph for $Y=3X+2$.
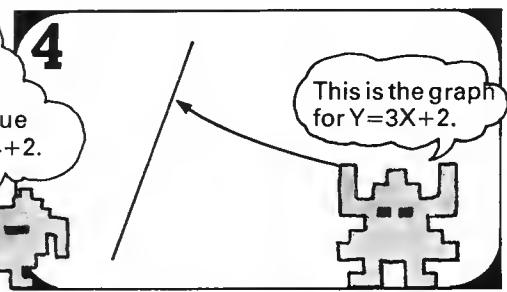
calculate Y and line 30 plots X and Y on the screen. In graphs programs, you must make sure the maximum values for X and Y will fit on the screen or you will get a bug.

Computers and maths

In calculations which have several parts, such as $3\times X+2$, the computer always does the multiplications or divisions before it adds or subtracts. This means that the computer would give the same answer for these two sums:

```
PRINT 4*6+8      PRINT 8+4*6
32               32
```

If you want the computer to do the sum in a different order you use brackets, like this:

```
PRINT (8+4)*6
72
```

This time the computer adds 8 and 4, then multiplies by 6.

Program puzzle

```
THINK OF A NUMBER
DOUBLE IT, ADD 4
DIVIDE BY 2, ADD 7
MULTIPLY BY 8, SUBTRACT 12
DIVIDE BY 4 AND TAKE AWAY 11
TELL ME THE RESULT.
THE NUMBER YOU FIRST THOUGHT
OF IS ....
```
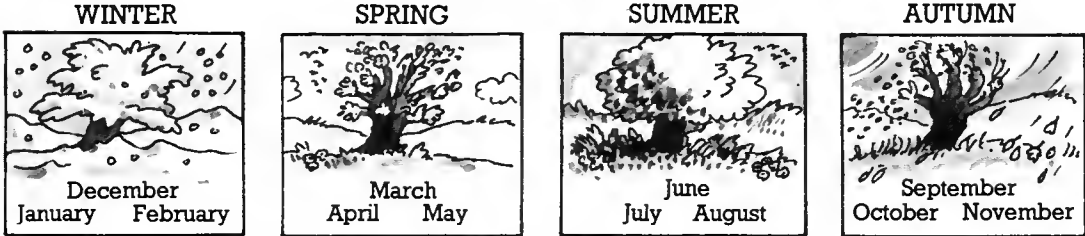
Can you write a program to get the computer to carry out this well known number trick? (To find the number you first thought of you subtract 4 from the result, then divide by 2.)

Birthdays program

This program uses another way to display information on the screen. It uses symbols to compare the number of people who were born in different seasons of the year. You could use a program like this to compare, say, sightings of a certain bird in different seasons, or the number of wins of different football teams. Before writing a long program like this it is a good idea to write a program plan.

Program plan

Aim: To compare the number of people with birthdays in winter, spring, summer and autumn.

WINTER	SPRING	SUMMER	AUTUMN

December January February	March April May	June July August	September October November

1. Give the computer the data (i.e. the seasons when the people were born) for a survey of 20 people.
2. Store the data in the computer.
3. Present the data on the screen.

Sample run

```
RUN
WINTER  TOTAL*******
SPRING  TOTAL***
SUMMER  TOTAL*****
AUTUMN  TOTAL*****
```

The program

```
5 LET A=0
6 LET B=0
7 LET C=0
8 LET D=0
```
Empty variables ready to use for running totals for each season.

```
10 FOR I=1 TO 20
20 PRINT "PERSON ";I;" WAS BORN IN"
30 PRINT "WINTER, SPRING, SUMMER OR AUTUMN"
40 PRINT "TYPE W, SP, SU OR A"
50 INPUT B$
```
Loop to make computer ask question once for each person in survey.

```
60 IF B$="W" THEN LET A=A+1
70 IF B$="SP" THEN LET B=B+1
80 IF B$="SU" THEN LET C=C+1
90 IF B$="A" THEN LET D=D+1
```
Lines 60 to 100 check the answer in B$ and add one to the variable for that season.

```
100 NEXT I
```
Sends computer back to repeat question.

```
110 PRINT "WINTER TOTAL";
115 LET N=A
120 GOSUB 200
130 PRINT "SPRING TOTAL";
135 LET N=B
140 GOSUB 200
150 PRINT "SUMMER TOTAL";
155 LET N=C
160 GOSUB 200
170 PRINT "AUTUMN TOTAL";
175 LET N=D
180 GOSUB 200
190 STOP
200 REM: SUBROUTINE TO PRINT STARS
210 IF N=0 THEN GOTO 250
220 FOR I=1 TO N
230 PRINT "*";
240 NEXT I
250 PRINT
260 RETURN
```

The subroutine makes the computer print a number of stars equal to the number in each variable.
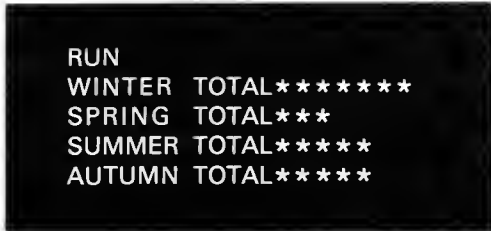
By putting the total into N each time, the computer can use the same routine for each season.

Makes the computer stay on the same line to print the stars.

Line 210 checks in case no-one was born in a particular season.

The main program sets N to the total for A, B, C or D. The loop makes the computer carry out line 230 "N" times.

83

More graphics

These two pages show how you can use PLOT and UNPLOT to make moving pictures on the screen. Moving pictures are called animated graphics and they are useful for games programs, or to illustrate programs which explain, say, the principles of gravity or ballistics and flightpaths.
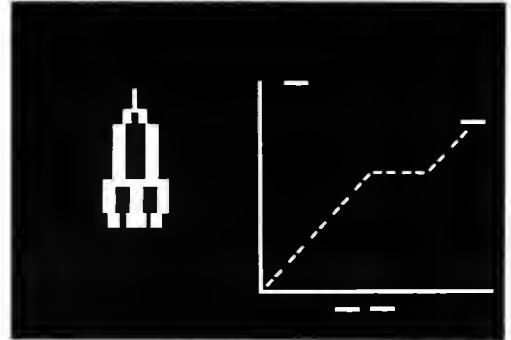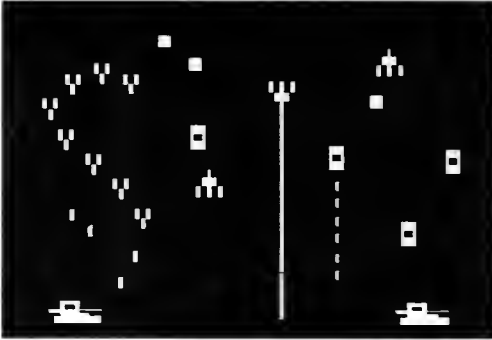
The pictures for video and arcade games are controlled by a small computer. The computer is programmed to play only the games and the programs are in the computer's own code, not in BASIC.

A general purpose microcomputer programmed in BASIC makes slower, simpler pictures. It cannot handle all the instructions for the screen quickly enough to make really fast moving graphics.

Plot/unplot program

1

```
10 LET X=1
20 LET Y=1
30 PLOT (X,Y)
40 UNPLOT (X,Y)
50 LET X=X+1
60 LET Y=Y+1
70 GOTO 30
```

Some computers need a graphics mode line.

This short program makes a spot of light move across the screen. Remember, the commands for PLOT and UNPLOT vary on different computers.

2

This is how the spot moved. In a moment this spot will be unplotted too.
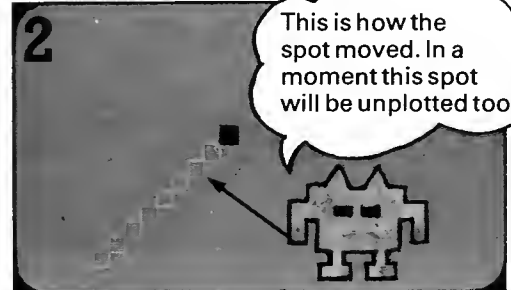
When the spot reaches the edge of the screen the program may stop with an error message as the values for X and Y are outside the screen range of the computer.

3

LET X=X+1

LET Y=Y−1

LET Y=Y+1

LET X=X−1

Bat and ball video games use programs like the one above to move the ball on the screen. There are simple program rules to keep the ball moving when it reaches the edge of the screen.

When the ball reaches the top of the screen the amount to be added to Y is subtracted instead. In the same way, when it reaches the right edge, the amount is subtracted from X.

Line pattern program

This program plots a line across the screen and when it reaches the sides, sends it back again in another direction. It does not use UNPLOT so the lines leave a pattern on the screen. The picture on the right shows what happens when you run the program. The program is set by line 100 to plot 10,000 pixels. You can change this figure to make it shorter, or BREAK the program at a pattern you like.
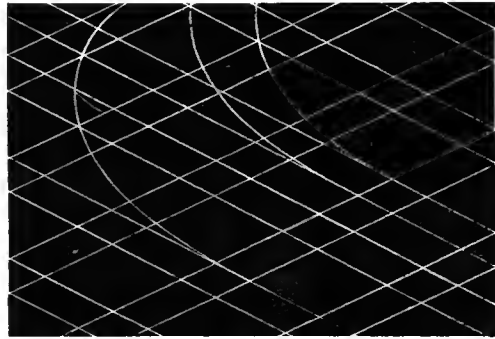

```
 10  REM: SET UP GRAPHICS MODE HERE IF NECESSARY
 20  PRINT "HOW MANY PIXELS ACROSS?";
 30  INPUT H
 40  PRINT "AND UP?";
 50  INPUT V
 55  CLS
 60  LET X=H/2
 70  LET Y=V/2
 80  LET S=1
 90  LET T=1
100  FOR I=1 TO 10000
110  LET S=S+(INT(RND(1)*10+1)-5)/50
120  LET X=X+S
130  LET Y=Y+T
140  IF X<5 THEN LET S=-S
150  IF X>H-5 THEN LET S=-S
160  IF Y<5 THEN LET T=-T
170  IF Y>V-5 THEN LET T=-T
180  GOSUB 300
190  NEXT I
200  STOP
300  REM: PLOT LINE
310  PLOT (X,Y)
320  RETURN
```

Lines 20 to 50 ask for the height and width of the screen. The semi-colon puts your reply on the same line as the question.

This makes X and Y start at the centre of the screen.

S and T are the amounts that will be added to X and Y to make the line move.

The loop from lines 100 to 190 is repeated 10,000 times Each time, X and Y are changed by a small amount.
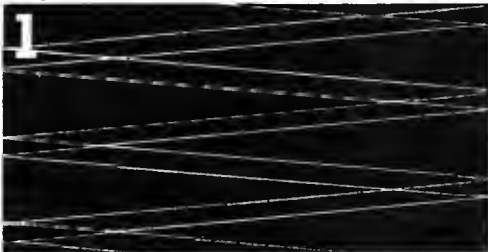
This gives a very small number to add to X. The number varies each time the loop is repeated.

These lines test for the edges and reverse S and T when X and Y come within five pixels of an edge.
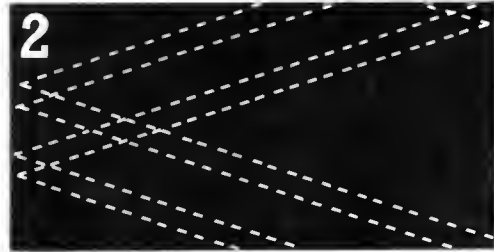
Sends the computer to the subroutine to plot the line

Plots the pixel with the current value for X and Y.

Experiments

Line 110 adds a very small random amount to X each time and this makes the line wiggle across the screen. If you have a computer, try deleting this line. The lines on the screen should become parallel.
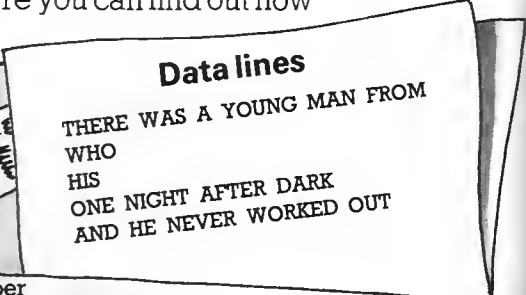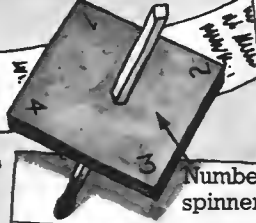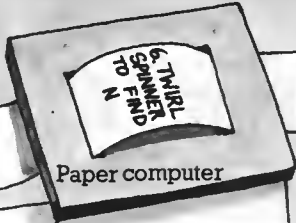
Try changing the numbers in lines 80 and 90 to; say, 5 or 10 (or larger on a computer with high resolution graphics). This makes the computer plot the pixels at intervals.

85

Funny poems program

The next few pages show you how to write a program which can compose lots of poems. A version of this program first appeared in the *Usborne Guide to Computers*. That book showed how to make a "paper computer" which used a simple version of this program. Here you can find out how to write the same program in BASIC.

Program for the paper computer

Paper computer

Number spinner

Data lines

THERE WAS A YOUNG MAN FROM

WHO

HIS

ONE NIGHT AFTER DARK

AND HE NEVER WORKED OUT

Data words

TASHKENT	TRENT	KENT	GHENT
WRAPPED UP	COVERED	PAINTED	FASTENED
HEAD	HAND	DOG	FOOT
IN A TENT	WITH CEMENT	WITH SOME SCENT	THAT WAS BENT
IT RAN OFF	IT GLOWED	IT BLEW UP	THAT WAS BENT
IN THE PARK	LIKE A QUARK	FOR A LARK	IT TURNED BLUE
WHERE IT WENT	ITS INTENT	WHY IT WENT	WITH A BARK
			WHAT IT MEANT

1 A=0 and B=0
2 Add 1 to A
3 If A=6 go to line 10
4 Write data line A
5 Add 1 to B
6 Twirl spinner to find N
7 Write data words from row B column N
8 If B=3 or 5 go to line 5
9 Go to line 2
10 Stop

There was a young man from Kent
Who wrapped up his head in cement
One night after dark
It turned blue in the park
And he never worked out where it went

This is the program for the paper computer. It looks a little like BASIC, but it would not work on a real computer. The words and phrases for the poem are "stored" on pieces of paper and the program tells you which to select. The number spinner is a random number generator to give random numbers between one and four.

1 Translating the program into BASIC

This won't work on a computer yet.


```
10  LET A=0
20  LET B=0
30  LET A=A+1
40  IF A=6 THEN STOP
50  Write data line A
60  LET B=B+1
70  LET N=INT(RND(1)*4+1)
80  Write data words from row B
    column N
90  IF B=3 THEN GOTO 60
100 IF B=5 THEN GOTO 60
110 GOTO 30
120 END
```

These lines set up empty variable spaces.

Lines 30 and 40 keep count of the number of data lines the computer has selected.

Lines 50 and 80 are not in BASIC yet.

Line 60 keeps count of the number of data words.

Gives a random number between 1 and 4

Lines 90 and 100 send it back to select another data line.

Most of the program is easy to translate into BASIC, but lines 50 and 80 are more difficult. The computer needs a way of storing and picking out the data lines and words which are needed for each line of the poem.

2 Giving the computer data

```
50 READ A$
:
180 DATA THERE WAS A YOUNG MAN FROM,
    WHO, HIS
190 DATA ONE NIGHT AFTER DARK, AND HE
    NEVER WORKED OUT
```
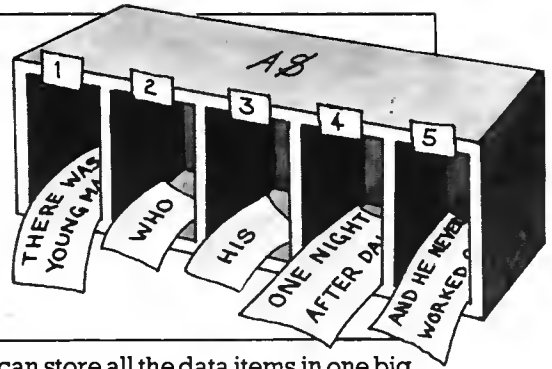

To give the computer the data lines and words you can use READ . . . DATA. Each time the computer carries out the READ instruction it takes another item from the DATA line and stores it in the variable.

You can store all the data items in one big variable called A$. A variable containing more than one data item is called an array and each item is referred to by a number, e.g. READ A$(3) gives HIS.*

3
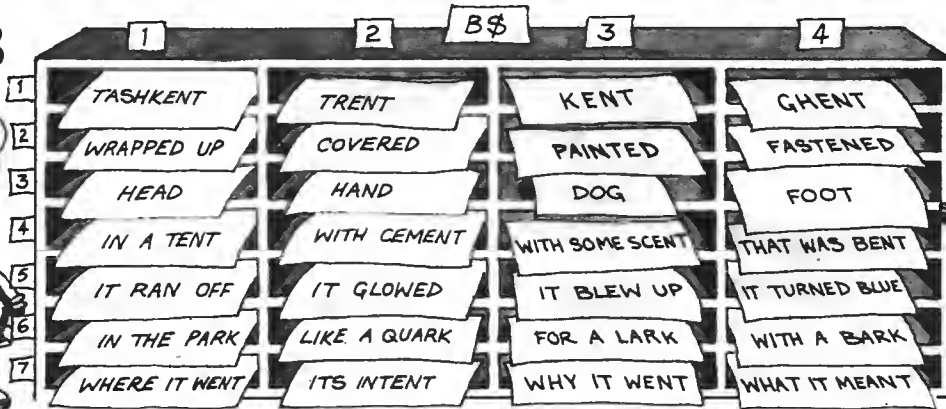
A variable can also hold several rows of data and you can store all the data words in a variable like this. It is called a two-dimensional array. Here, each data item is referred to by the number of the row and column it is in. So READ B$(4,2) gives WITH CEMENT and READ B$(6,3) gives FOR A LARK. You can store numbers in arrays, too, using a number variable, e.g. N(5,7).

4 Putting the data in the variables


```
10 FOR I=1 TO 7 }——— I is the row number
20 FOR J=1 TO 4 }——— J is the column
30 READ B$(I, J)            number
40 NEXT J
50 NEXT I
60 DATA TASHKENT, TRENT, KENT, GHENT
70 DATA WRAPPED UP, COVERED, PAINTED, FASTENED
80 DATA HEAD, HAND, DOG, FOOT
90 DATA IN A TENT, WITH CEMENT, WITH SOME SCENT, THAT WAS BENT
100 DATA IT RAN OFF, IT GLOWED, IT BLEW UP, IT TURNED BLUE
110 DATA IN THE PARK, LIKE A QUARK, FOR A LARK, WITH A BARK
120 DATA WHERE IT WENT, ITS INTENT, WHY IT WENT, WHAT IT MEANT
```
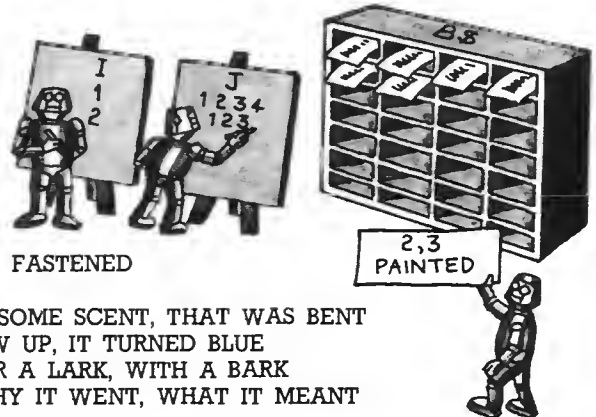
To read each data item into the variable you need to be able to alter the numbers in brackets after READ. You can do this with loops. B$ needs nested loops as shown above with an I loop for the row number

and a J loop for the column number. Each time the I loop is carried out the J loop is repeated four times – once for each of the columns in a row.

87

*Sinclair computers deal with variables in a different way and this program will not run on a Sinclair. You can find out more about this over the page.
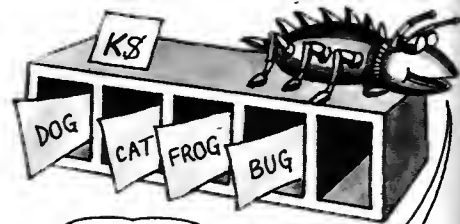
5 Making space for variables

```
5 DIM K$(5)          ⎫──── This is the size of the variable,
10 FOR I=1 TO 5            i.e. 5 items in a row.
20 READ K$(I) ⎤────── This line puts the data in K$
30 NEXT I                  each time the loop is repeated.
40 STOP

60 DATA DOG, CAT, FROG, BUG
```

This program has a bug.
There are not enough data
items for K$.

At the beginning of the program you have to tell the computer how big you want the variable to be. You do this with the word DIM followed by the variable name and the number of data items, e.g. DIM K$(5).

For a two dimensional array you give the computer the number of rows and columns in the variable, e.g. DIM C$(5,3). You must always have the right number of data items for the variable or you get a bug.

6 Printing out the data

```
200 LET A=0
210 LET B=0
220 LET A=A+1 ⎤──────
230 IF A=6 THEN STOP
240 PRINT A$(A)
250 LET B=B+1 ⎤──────
260 LET N=INT(RND(1)*4+1)
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250 ⎤────
290 IF B=5 THEN GOTO 250 ⎦
300 GOTO 220 ⎤──────
310 END
```

A keeps count of the number of times this section of the program is repeated.

B keeps count of the data word rows and makes sure that the correct row is used with each data line.

Lines 280 and 290 make the computer print out words from another data word row before printing the next data line.

This sends the computer back to print the next data line.

The computer needs these lines to print out the data lines and words in the right order. This section of the program is repeated five times. Each time, the computer prints out data line number A and some data words from row number B. The actual data words which are chosen are decided by random number N.

Sinclair computers and variables

This program does not work in its present form on Sinclair computers because they handle strings in a different way.

A$(7 TO 9) is DOG.

For two-dimensional arrays you have to tell the computer the number of the row as well as the numbers of the characters. For instance, A$(2, 4 TO 6) is PUG.

To tell a Sinclair computer to pick out a particular data item from a variable you have to give it the numbers of the first and last characters of the item you want. This is the same system as the Sinclairs use for LEFT$, RIGHT$, etc. (See page 80.)
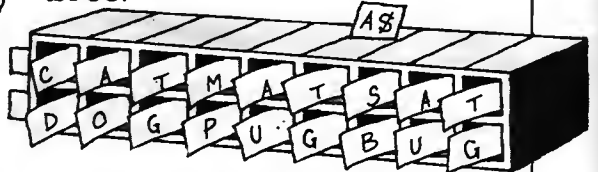
At the beginning of a program you tell the computer how many rows the array has, and how many characters there are in each row, e.g. DIM A$(2,9) means two rows, each with nine characters. All the rows in the array must have the same number of characters.

The complete funny poems program

Now you can put the parts of the program together and write the complete poetry program. The first part of the program (lines 10 to 190) give the computer the data and the second part (lines 200 to 310) prints out the poem. Each time you run the program you get a different version of the poem because the random number N makes the computer pick different words.

```
 10 DIM A$(5)
 20 DIM B$(7,4)
 30 FOR I=1 TO 7
 40 FOR J=1 TO 4
 50 READ B$(I, J)
 60 NEXT J
 70 NEXT I
 80 DATA TASHKENT, TRENT, KENT, GHENT
 90 DATA WRAPPED UP, COVERED, PAINTED, FASTENED
100 DATA HEAD, HAND, DOG, FOOT
110 DATA IN A TENT, WITH CEMENT, WITH SOME SCENT, THAT WAS BENT
120 DATA IT RAN OFF, IT GLOWED, IT BLEW UP, IT TURNED BLUE
130 DATA IN THE PARK, LIKE A QUARK, FOR A LARK, WITH A BARK
140 DATA WHERE IT WENT, ITS INTENT, WHY IT WENT. WHAT IT MEANT
150 FOR I=1 TO 5
160 READ A$(I)
170 NEXT I
180 DATA THERE WAS A YOUNG MAN FROM, WHO, HIS
190 DATA ONE NIGHT AFTER DARK, AND HE NEVER WORKED OUT
200 LET A=0
210 LET B=0
220 LET A=A+1
230 IF A=6 THEN STOP
240 PRINT A$(A)
250 LET B=B+1
260 LET N=(RND(1)*4+1)
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250
290 IF B=5 THEN GOTO 250
300 GOTO 220
310 END
```

Lines 10 and 20 tell the computer how much space to leave for the variables – a row of 5 for A$ and 7 rows of 4 for B$.

These are the nested loops for putting the data in B$.

Lines 80 to 140 contain all the data words to be stored in B$.

This is a loop to put the data into A$.

Lines 180 to 190 contain all the data lines to be stored in A$.

This prints the data line stored in A$ compartment number A.

This prints the data words stored in B$ row B, column N.

The program stops at line 230 when A=6, so it never reaches line 310, but some computers need an END anyway.

Sample runs

```
THERE WAS A YOUNG MAN FROM        THERE WAS A YOUNG MAN FROM
KENT                             GHENT
WHO                              WHO
WRAPPED UP                       PAINTED
HIS                              HIS
HEAD                             FOOT
IN A TENT                        WITH CEMENT
ONE NIGHT AFTER DARK             ONE NIGHT AFTER DARK
IT GLOWED                        IT TURNED BLUE
LIKE A QUARK                     WITH A BARK
AND HE NEVER WORKED OUT          AND HE NEVER WORKED OUT
WHY IT WENT                      ITS INTENT
```

Here are two of the 16,384 possible different versions of the poem. If you try this program and always get the same poems, look in your manual for how to make the computer produce different random numbers. Some computers produce the same sequence of random numbers each time they are switched on.

Programming tips

On these two pages there are some tips to help you write your own programs, and a list of the most common bugs you might get, and what causes them. The most likely bugs are listed first, so if you have a program which will not work, check through this list until you find the reason.
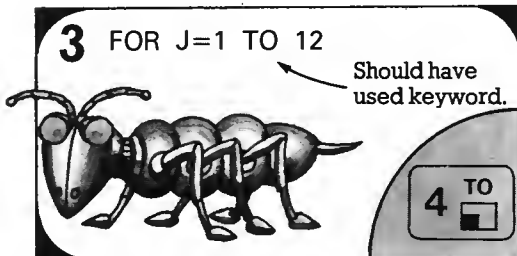
Finding bugs

1 PRONT ← Misspelt word
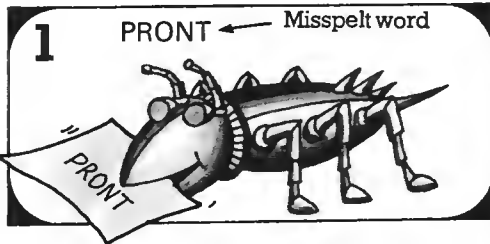
Look for typing mistakes in BASIC words. If you misspell one of these words the computer will not recognize it.

2 FOR 1=1 TO 6

Figure 1 instead of letter I.
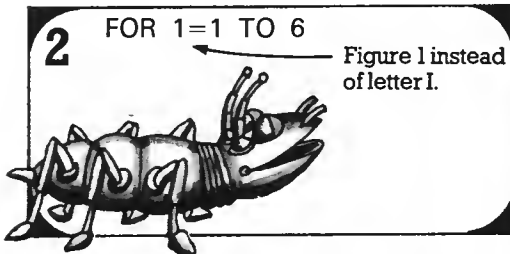
Check Os and 0s and 1s and Is to make sure you have typed the right ones in the right places.

3 FOR J=1 TO 12

Should have used keyword.

4 TO

If you have a Sinclair computer, make sure you have not typed a word in letter by letter instead of pressing the key for that word.

Writing programs

When you are writing programs it helps to remember that the computer can carry out three main activities: simple instructions, repeating things and making decisions. These are the building blocks of all programs.

SIMPLE INSTRUCTIONS

```
LET A=3
LET N=N+1
PRINT A/T
PLOT (X,Y)
```

REPEATING THINGS

```
FOR J=1 TO 6

20 LET A=1
30 IFA< 10 THEN
      GOTO 100
```

MAKING DECISIONS

```
IF X=Y THEN STOP
IF K$="HELLO"
THEN PRINT A
```

This book has covered all the main instructions you need in BASIC to tell the computer to carry out these activities. When you are writing a program, work out what the computer needs to do at each stage, then decide which instructions you want to use.

4 Missing quotes

```
PRINT "SHOESTRING
DATA ONE, TWO THREE
```

Missing comma
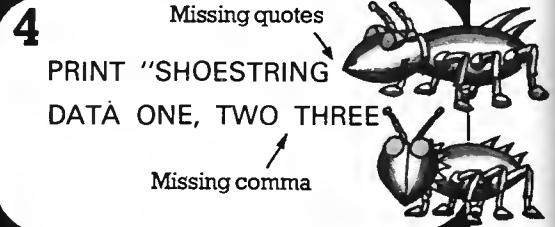
Make sure you have not left out any quotation marks, or the commas between data items. Check complicated lines which have lots of symbols especially carefully.

There are usually several different ways to write a program and some of them may be neater and shorter than others. When you are writing a long program it is a good idea to divide it up into lots of sections with subroutines to carry out each activity. The central core of the program may be a simple set of instructions, decisions and repeats which controls when and how often the computer carries out the subroutines.

Calculating

Getting more information

Plotting

Keeping count
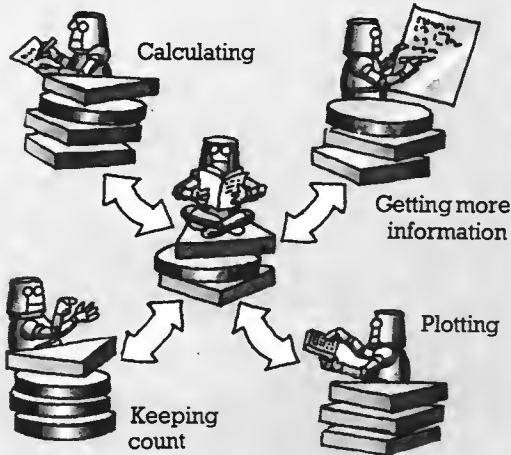
Breaking up programs into sections like this makes it much easier to find any mistakes. Each section can usually be tested by itself without running through all the program. Remember to label each section with a REM line so you know what it is for.

5 PRINT RND (4)

DON'T UNDERSTAND

PLOT (X,Y)

WHAT?
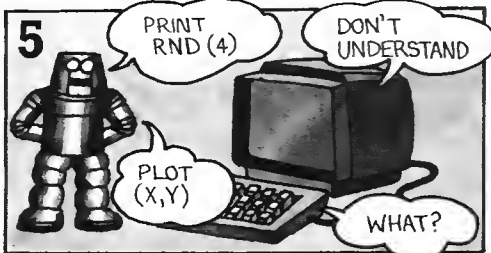
Make sure you use the correct RND, PLOT and CLS commands for the computer. Check, too, that you have given the computer a general graphics line if it needs one.

Error messages

All computers print out some sort of message when there is a bug in the program and the messages are explained in the computer's manual. Here are some of the most common messages you may get.

Out of data

◀ This means there are not enough data items for the computer to read in the DATA lines. It may be because you have missed out a comma between two items, so the computer has read them as one.

▶ The line with the number given in a GOTO or GOSUB statement does not exist. You may have accidentally erased the line by typing in another line with the same number, or you may have just mistyped the number.

No such line

No such variable

◀ You may get this report on a BBC or Sinclair computer. It usually means you have not set up a variable with a line such as LET C=0 or LET C="" before using it.

▶ This means the NEXT line of a loop is missing. It may be because you typed the wrong variable name, or even put a 1 instead of an I so the computer did not recognize it.

FOR without NEXT

Last word

Some bugs are very hard to find, but if the computer will not run the program there must be a bug in it somewhere. If you really cannot find the bug, try typing in suspect or complicated lines again, you might get them right the second time without even noticing what the bug was.

Puzzle answers

Page 63
Name and message program

```
10 PRINT "WHAT IS YOUR NAME"
20 INPUT N$
30 PRINT "HELLO"
40 PRINT N$
50 PRINT "HOW ARE YOU"
```

Page 65
1. Sums program

```
10 LET A=9
20 LET B=7
30 PRINT A*B
40 PRINT A/B
50 LET A=A+1
60 LET B=B+3
70 PRINT A*B,A/B      Comma to
80 END                 leave space
```

2. Tables program   Spaces

```
30 PRINT A;" TIMES ";B;" IS ";A*B
40 PRINT A;" DIVIDED BY ";B;" IS ";A/B
```

3. Name and message alterations

```
10 PRINT "WHAT IS YOUR NAME"
20 INPUT N$
30 PRINT " HELLO ";N$;" HOW
   ARE YOU
```

Page 66
Sums program

```
10 PRINT "WHAT IS 7 TIMES 7"
20 INPUT A
30 IF A=49 THEN PRINT "CORRECT"
40 IF A<>49 THEN PRINT "NO";7*7
```

You need a semi-colon after the
quotes, like this.

Page 67
Age guessing game

Replace line 30 and add a new line 35:

```
30 IF G<14 THEN PRINT "OLDER
   THAN THAT"
35 IF G>14 THEN PRINT "YOUNGER
   THAN THAT"
```

Page 71
Plotting counter

```
 5 LET C=0
45 LET C=C+1
50 IF C<6 THEN GOTO 10
```

Plotting your initial

Here is an example of a program to plot
the letter L.

```
10 LET X=15
20 LET Y=30
30 PLOT (X,Y)
40 LET Y=Y-1
50 IF Y>5 THEN GOTO 30
60 LET X=X+1
70 PLOT (X,Y)
80 IF X<45 THEN GOTO 60
90 END
```

Page 72
Random numbers

The formula for a random number
between 10 and 20 would be
INT(RND(1)*11+9). On computers which
need only a number in brackets after
RND, it would be RND(11)+9. There are
eleven possible numbers between 10 and
20 so you need to pick random numbers
between 1 and 11, then add 9.

Page 73
Space attack

These are the lines you need to add to
count the number of hits:

```
15 LET S=0
75 IF X=A*B THEN LET S=S+1
95 PRINT "YOU HIT "
   ;S;" OUT OF 6 ALIENS"
```

Page 75
1. Eight times table

```
10 PRINT "THE EIGHT TIMES TABLE"
20 FOR J=1 TO 12
30 PRINT J;"x8 = ";J*8
40 NEXT J
```

2. N times table

```
10 INPUT "TYPE IN A NUMBER";N
20 PRINT "HERE IS THE "
   ;N;" TIMES TABLE"
30 FOR I=1 TO 12
40 PRINT I;" TIMES ";N;" IS ";I*N
50 NEXT I
60 INPUT "ANOTHER NUMBER (Y or N)"
   ;M$
70 IF M$="Y" THEN GOTO 10
```

For the ZX81 you need separate PRINT
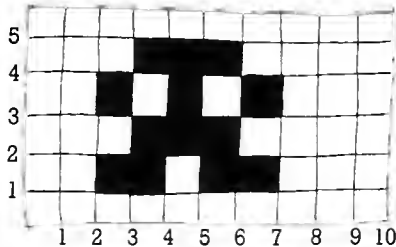and INPUT lines.

Computer book string puzzle
LEFT$(A$,8) is "COMPUTER"
RIGHT$(A$,10) is "PUTER BOOK"
MID$(A$,5,8) is "UTER BOO"

Space invaders repeat program

Number trick program

```
10 PRINT "THINK OF A NUMBER"
20 PRINT "DOUBLE IT, ADD 4"
30 PRINT "DIVIDE BY 2, ADD 7"
40 PRINT "MULTIPLY BY 8,
   SUBTRACT 12"
50 PRINT "DIVIDE BY 4 AND TAKE
   AWAY 11"
60 PRINT "TELL ME THE RESULT"
70 INPUT N
80 PRINT "THE NUMBER YOU FIRST
   THOUGHT OF IS";(N-4)/2
```

You need the brackets
to make the computer
do the sum in the order
you want.

1


```
5
4
3
2
1
  1 2 3 4 5 6 7 8 9 10
```

Draw a simple space invaders shape on
squared paper.

2

4,5; 5,5; 6,5

3,4; 5,4; 7,4

4,3; 5,3; 6,3

3,2; 4,2; 6,2; 7,2

Then work out the co-ordinates of all the
squares which make up the space
invader.

3

```
5 CLS
50 INPUT "HOW MANY POINTS ACROSS THE SCREEN";W
60 INPUT "HOW MANY UP";V
65 CLS
70 FOR I=0 TO V STEP V/6
80 FOR J=0 TO W STEP W/6



130 NEXT J
140 NEXT I
150 END
```

Change the 6 to a higher figure to increase the number of
times the invader shape repeats on the screen. (If you
get a bug you have made the number too big.)

Put your plot lines here, e.g.
90 PLOT (J+3, I+2)
92 PLOT (J+4, I+2)
for the two bottom left-hand squares of the
space invader shown above. You need a
program line for each square.

Copy out the pattern repeat program,
excluding lines 10 to 40 and 90 to 140, as
shown above. (These lines produce the
random pattern for the program so you do
not need them.)
Now you need to put your own plot lines

into the program between lines 80 and 140
(you can renumber the lines in the
program). For each pair of co-ordinates
you need to add J to the first figure and I to
the second figure, to make the space
invader repeat.

93

Loading and saving programs

Loading and saving programs on cassette is quite tricky and you can waste many hours trying to load a program you have saved. Below there are descriptions of some of the problems you may encounter, with advice on how to solve them.

Loading programs

To load a program successfully from tape you need to find the correct volume setting on your cassette recorder. If the output from the recorder is too strong, or weak, the computer will not be able to understand the signals.

Finding the correct volume setting is just a matter of trial and error. If your recorder has a tone dial, set this to maximum treble and leave it there. Set the volume control to a middle position. Follow the computer's instructions for loading programs and press PLAY on the recorder. Most computers give some indication to tell you when the program has loaded, but sometimes you may load a program successfully and then find it is full of bugs because the output from the recorder was distorted.

If you did not manage to load the program, rewind the tape, type NEW on the computer to clear out its memory, and then try again with a slightly different volume setting. After several attempts you should find the correct volume level for your computer.

If you still have problems, though, try cleaning the heads of the cassette with a cleaning tape. You can buy one of these from a hi-fi shop. If this does not help, it may be that the program you are trying to load is badly recorded on the tape. If it is a program you have saved yourself, read the section opposite on saving programs. If it is a tape you have bought ask someone else, if possible, to try it on their equipment, and if they cannot load it return the tape to where you bought it.

After finding the correct volume setting you should be able to use the same setting to load all the programs you save yourself, but bought tapes and those recorded by other people may need slightly different settings.

Saving programs

When you are saving a program you do not have to worry about the volume setting on the recorder. Just follow the computer's instructions for saving programs and press RECORD on the cassette recorder. It is a good idea to record the same program several times.

After saving the program you can play it back like an ordinary tape (pull the "ear" lead out of the recorder so that its loudspeaker works). You should hear a series of high-pitched bleeps. Now try loading the program to make sure it works. If it is a bad copy you will not be able to load it, or you may load it and find it is full of bugs.

If you have a bad recording it may be because the tape is old and crinkled or the heads of the recorder may be dirty. Clean the heads with a cleaning tape and try saving the program again on different tape. You will probably have lost the program in the computer's memory and will have to type it in again.

If you still cannot load the program, the output from the computer when you saved the program may have been too strong, or weak, for your cassette recorder.

Most inexpensive cassette recorders have an automatic recording level control so there is nothing you can do to adjust the level of the signals going into the recorder. The easiest solution is to try and borrow a different make of cassette recorder and see if this works with your computer. You could also try asking a local user group or dealer for advice. If you take the computer and cassette recorder to the dealer, they will check the equipment and may be able to modify either the cassette or the computer so that they are compatible.

If your cassette recorder does have a recording level control, try saving the program lots of times at different recording levels until you find the correct setting.

ASCII chart

Code number	ASCII character	Code number	ASCII character
32	space	62	>
33	!	63	?
34	"	64	@
35	#	65	A
36	$	66	B
37	%	67	C
38	&	68	D
39	'	69	E
40	(	70	F
41	)	71	G
42	*	72	H
43	+	73	I
44	,	74	J
45	–	75	K
46	.	76	L
47	/	77	M
48	0	78	N
49	1	79	O
50	2	80	P
51	3	81	Q
52	4	82	R
53	5	83	S
54	6	84	T
55	7	85	U
56	8	86	V
57	9	87	W
58	:	88	X
59	;	89	Y
60	<	90	Z
61	=		

ZX81 code chart

Code number	ZX81 character	Code number	ZX81 character
11	"	41	D
12	£	42	E
13	$	43	F
14	:	44	G
15	?	45	H
16	(	46	I
17	)	47	J
18	>	48	K
19	<	49	L
20	=	50	M
21	+	51	N
22	–	52	O
23	*	53	P
24	/	54	Q
25	;	55	R
26	,	56	S
27	.	57	T
28	0	58	U
29	1	59	V
30	2	60	W
31	3	61	X
32	4	62	Y
33	5	63	Z
34	6		
35	7		
36	8		
37	9		
38	A		
39	B		
40	C		

Chart of screen sizes

	Max. number of characters across (or number of columns)	Max. number of lines down (or number of rows)
VIC 20	22	23
TRS-80	64	16
BBC	20/40/80	16/24/32
ZX81	32	22
ZX Spectrum	32	22
Apple	40	25

Conversion chart

This quick reference chart shows some of the variations in the BASIC used by different computers. It doesn't include instructions for graphics, sound or colour as these vary so enormously from machine to machine. Note also that although most computers (except the BBC) use PEEK and POKE, they do not use the same system of memory addresses, so the numbers used with PEEK and POKE must be changed for each computer.

	BBC	VIC/Pet	Apple	TRS-80	ZX Spectrum	ZX81
Select random number between 0 and 0.9999999	RND(1)	RND(1)	RND(1)	RND(0)	RND	RND
Select random number between 1 and N	RND(N)	RND(1)*N+1	RND(1)*N+1	RND(N)	RND*N+1	RND*N+1
Select random letter between A and Z	CHR$(RND(26)+64)	CHR$(INT(RND(1) *26+65))	CHR$(INT(RND(1) *26+65))	CHR$(RND(26)+64)	CHR$(INT(RND*26+65))	CHR$(INT(RND*26+38))
Clear screen	CLS	PRINT CHR$(147)	HOME	CLS	CLS	CLS
Check keyboard to see if key being pressed	INKEY$(N)	GET X$	X$="" IF PEEK(−16384) >127 THEN GET X$	INKEY$	INKEY$	INKEY$
Convert characters into code numbers	ASC("X") (using ASCII code)	ASC("X") (using ASCII code)	ASC("X") (using ASCII code)	ASC("X") (using ASCII code)	CODE("X") (using ASCII code)	CODE("X") (using ZX81 code)
Move cursor right	PRINT CHR$(9)	PRINT CHR$(29)	PRINT CHR$(21)	PRINT CHR$(25)	PRINT CHR$(9)	PRINT CHR$(115)
Move cursor left	PRINT CHR$(8)	PRINT CHR$(157)	PRINT CHR$(8)	PRINT CHR$(24)	PRINT CHR$(8)	PRINT CHR$(114)
Move cursor down	PRINT CHR$(10)	PRINT CHR$(17)	PRINT CHR$(10)	PRINT CHR$(26)	PRINT CHR$(10)	PRINT CHR$(113)
Move cursor up	PRINT CHR$(11)	PRINT CHR$(145)	CALL −998	PRINT CHR$(27)	PRINT CHR$(11)	PRINT CHR$(112)
Take 1st N characters of string	LEFT$(A$,N)	LEFT$(A$,N)	LEFT$(A$,N)	LEFT$(A$,N)	A$(1 TO N)	A$(1 TO N)
Take last N characters of string	RIGHT$(A$,N)	RIGHT$(A$,N)	RIGHT$(A$,N)	RIGHT$(A$,N)	A$(N TO)	A$(N TO)
Take middle N characters of string	MID$(A$,N1,N2)	MID$(A$,N1,N2)	MID$(A$,N1,N2)	MID$(A$,N1,N2)	A$(N1 TO N2)	A$(N1 TO N2)

COMPUTER SPACEGAMES

Daniel Isaaman
and Jenny Tyler

Contents

This section of the book contains simple games programs to play on a microcomputer. They are written for use on ZX81, ZX Spectrum, BBC, VIC 20, TRS-80 and Pet and Apple micros, and many are short enough to fit into the ZX81's 1K of memory.

Most micros use the language BASIC, but they all have their own variations or dialects. In this section of the book, the main listing for each program works on the ZX81 and lines which need changing for the other computers are marked with symbols and printed underneath. The fact that the programs are written for several micros means that they do not make full use of each one's facilities. You could try finding ways of making the programs shorter and neater for your micro.

For each game, there are ideas for changing and adding to the programs and towards the back of the book you will find tips and hints on writing games of your own. If you want to adapt programs from magazines and other books to run on your micro, there is a conversion chart to help you on page 96.

Typing in the programs

Lines which need changing for computers other than ZX81 are marked with these symbols:
- ▲ VIC and Pet
- ★ BBC and Acorn Electron
- ▬ TRS-80
- ● Apple
- s ZX Spectrum

Every time you see the symbol for the micro you are using, look below for the corresponding line number with the same symbol and type that in instead.

VIC 20 versions of all except the graphics program should work on Pet computers.
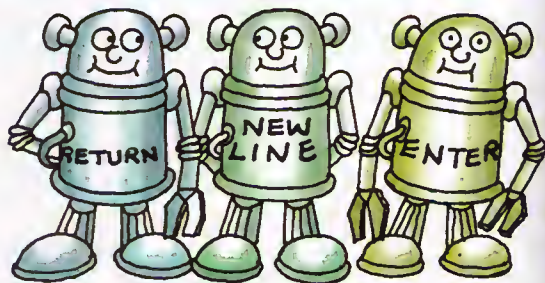
Points to remember

1 Type the lines exactly as they are printed including all punctuation and spaces.

2 Press RETURN, NEWLINE or ENTER key at the end of each program line.

3 Check each line as you go.

4 Make sure you don't miss out a line or confuse one with another. A piece of paper or a ruler is useful to mark your place in the listing.

5 Look out for the symbols and make sure you use the correct lines for your computer.

6 If you are using a ZX81 or ZX Spectrum, remember not to type the program instructions letter by letter but to use the special key for each instruction instead.

You may find it easier to get someone to read the program out to you while you type it in. Don't forget to explain that they must read every comma, fullstop, bracket and space and differentiate between letter "O" and zero, FOR and 4, and TO and 2.

Debugging programs

When you have typed in the program, check your manual to find out how to display it on the screen. (Usually you type LIST followed by the line numbers of the section you want to see.)

Check you have typed everything correctly. It is easy to make mistakes, so expect to find some. Use your manual to find out how to make changes to the program once it is typed in. If in doubt, you can always type the line again. All the computers will replace an existing line with a new one with the same number.

Here is a checklist of common mistakes to look out for:

1 Line missed out

2 Line wrongly numbered

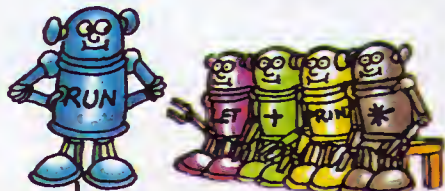3 The beginning of one line joined onto the end of another.
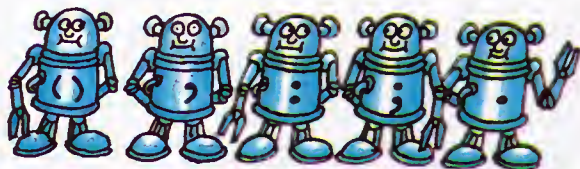
4 Brackets, commas, colons, semi-colons, fullstops or spaces missed out, especially in long, complicated lines. Watch for double brackets in particular.

5 Wrong line used for your computer.

6 Letter "O" confused with zero.

7 Wrong numbers used, e.g. extra zeros included.

Playing the games

To start the game you must type RUN. In some games things happen very quickly, so make sure you have read the instructions and know what you are supposed to do.

It is quite likely that the program still has a mistake in it and either won't run at all or the game won't work properly. Sometimes your computer will give you an error code which you can look up in the manual. This may help you find the mistake, though not always. List the program again and check it carefully against the book.

When the game is over, the computer will usually say something like BREAK IN LINE 200. To play again, you have to type RUN.

Experimenting with the games

There are suggestions for changing and adding to the programs throughout this section of the book, but don't be afraid to experiment with changes of your own. You can't damage the computer and you can always change back to the original if the changes don't work.

You will probably find you want to adjust the speed of some games, * especially after you have played them a number of times. You will find out which line to change on each program page.

Wherever you see PRINT, you can change the message in quotes that follows it to whatever you like. Also, unless you have ZX81 with only 1K, you can add extra messages.
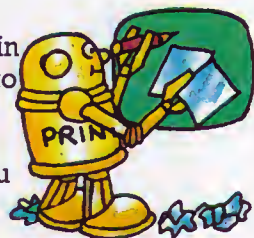
Type a line number (say 105 if you want to add a message between lines 100 and 110), then type PRINT, then your message inside quotes.

If your computer can make colours and sounds, you could use your manual to find out how they work and try adding them to the games in this book.
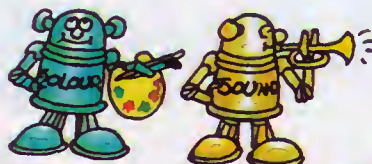
Starship Takeoff

You are a starship captain. You have crashed your ship on a strange planet and must take off again quickly in the alien ship you have captured. The ship's computer tells you the gravity on the planet. You must guess the force required for a successful take off. If you guess too low, the ship will not lift off the ground. If you guess too high, the ship's fail-safe mechanism comes into operation to prevent it being burnt up. If you are still on the planet after ten tries, the aliens will capture you.

```
▲●10 CLS
   20 PRINT "STARSHIP TAKE-OFF"
★■▲●30 LET G=INT(RND*20+1)
★■▲●40 LET W=INT(RND*40+1)
```

Clears the screen.

Computer selects two numbers – one between 1 and 20 to be put in G, the other between 1 and 40 to be put in W.
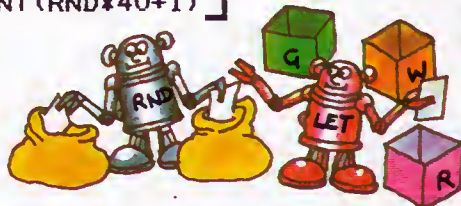
```
   50 LET R=G*W
```

Multiplies the number in G by number in W. Puts result in R.

```
   60 PRINT "GRAVITY= ";G
```

Prints GRAVITY and number in G.

```
   70 PRINT "TYPE IN FORCE"
```

Asks you for a number.

```
   80 FOR C=1 TO 10
```

This begins a loop which tells the computer to repeat the next section 10 times, to give you 10 goes.

```
   90 INPUT F
```

Stores your number in F.

```
  100 IF F>R THEN PRINT "TOO HIGH";
  110 IF F<R THEN PRINT "TOO LOW";
  120 IF F=R THEN GOTO 190
```

Compares number in F with number in R and prints appropriate message or jumps to 190.

```
  130 IF C<>10 THEN PRINT ", TRY AGAIN"
```

Prints if you've had less than 10 goes without a correct answer.

```
  140 NEXT C
```

End of loop. Goes back to 80 for another turn.

```
  150 PRINT
  160 PRINT "YOU FAILED -"
  170 PRINT "THE ALIENS GOT YOU"
  180 STOP
```

Prints after 10 unsuccessful goes.

```
  190 PRINT "GOOD TAKE OFF"
```

The above listing will work on a ZX81. For other computers, make the changes below.

On all but ZX81 and Spectrum, you can replace STOP with END. (Notice what happens when you do.)

```
   ●10 HOME
   ▲10 PRINT CHR$(147)
★▲●30 LET G=INT(RND(1)*20)
  ■30 LET G=INT(RND(0)*20)
★▲●40 LET W=INT(RND(1)*40)
  ■40 LET W=INT(RND(0)*40)
```

How to make the game harder

You can change the program to give you less than 10 goes. Do this by altering the last number in line 80 and the number in line 130. (They must be the same.)

Puzzle corner

You could change the range of possible forces. See if you can work out how.

Intergalactic Games

There is fierce competition among the world's TV companies for exclusive coverage of the First Intergalactic Games. Everything depends on which company wins the race to put a satellite into orbit at the right height.

You are the Engineer in charge of the launch for New Century TV. The crucial decisions about the angle and speed of the launching rocket rests on *your* shoulders. Can you do it?
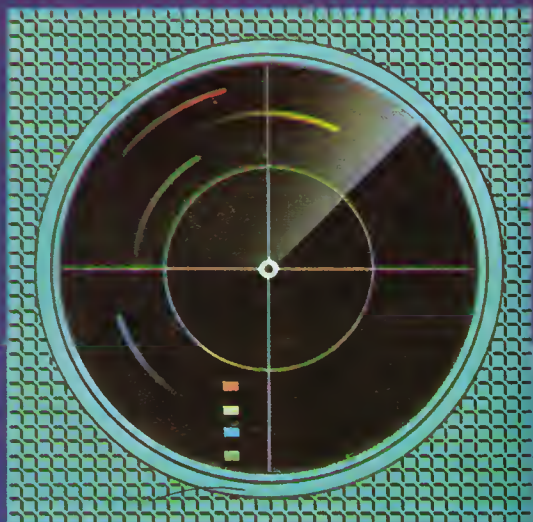
How the program works

```
  10 PRINT "INTERGALACTIC GAMES"
★■▲●20 LET H=INT(RND*100+1)
  30 PRINT "YOU MUST LAUNCH A SATELLITE"
  40 PRINT "TO A HEIGHT OF ";H

  50 FOR G=1 TO 8

  60 PRINT "ENTER ANGLE (0-90)"
  70 INPUT A

  80 PRINT "ENTER SPEED (0-40000)
  90 INPUT V

  100 LET A=A-ATN(H/3)*180/3.14159

  110 LET V=V-3000*SQR(H+1/H)

  120 IF ABS(A)<2 AND ABS(V)<100 THEN GOTO 200

  130 IF A<-2 THEN PRINT "TOO SHALLOW"
  140 IF A>2 THEN PRINT "TOO STEEP"
  150 IF V<-100 THEN PRINT "TOO SLOW"
  160 IF V>100 THEN PRINT "TOO FAST"

  170 NEXT G
  180 PRINT "YOU'VE FAILED"
  190 PRINT "YOU'RE FIRED"
  200 STOP
  210 PRINT "YOU'VE DONE IT"
  220 PRINT "NCTV WINS-THANKS TO YOU"
  230 STOP
```

Chooses the height to which you must launch your satellite, puts it in H and prints it.

Beginning of loop to give you 8 goes.

Asks you for an angle and puts it in A.

Asks you for a speed and puts it in V.

Uses H to calculate what the angle should be and subtracts this from your guess to find out how close you were.

Works out what the speed should be and subtracts it from your guess.

Checks if you were close enough to win and if so jumps to 210.

Prints an appropriate comment to help you with your next go.

Goes back for another go.

Prints after 8 unsuccessful goes.

Prints if you win.

The above listing will work on a ZX81. For other computers, make the changes below.

```
  ■20 LET H=INT(RND(0)*100+1)
★▲●20 LET H=INT(RND(1)*100+1)
```

102

103
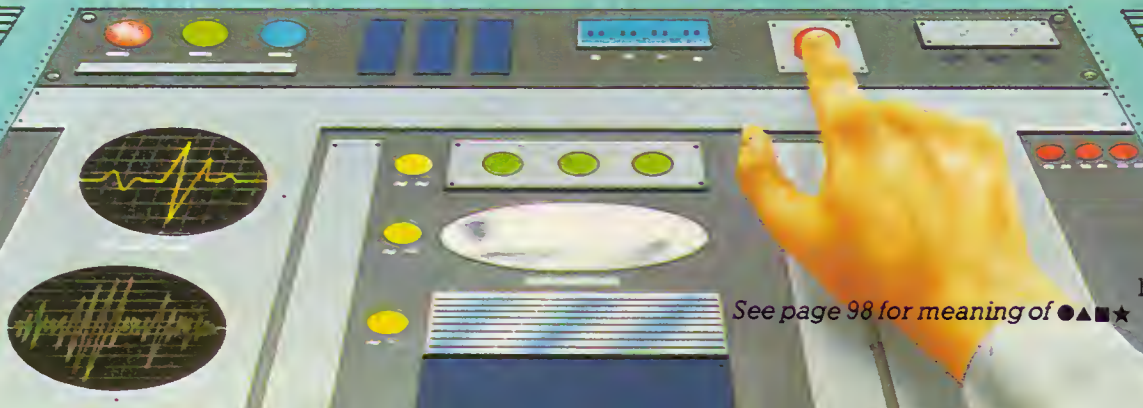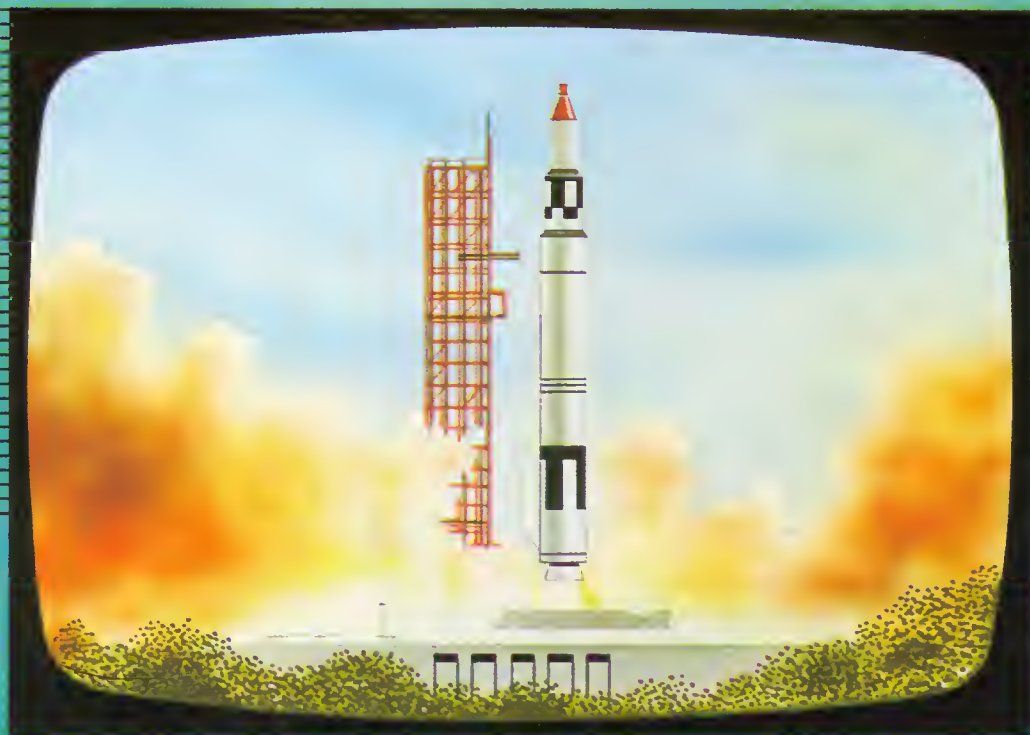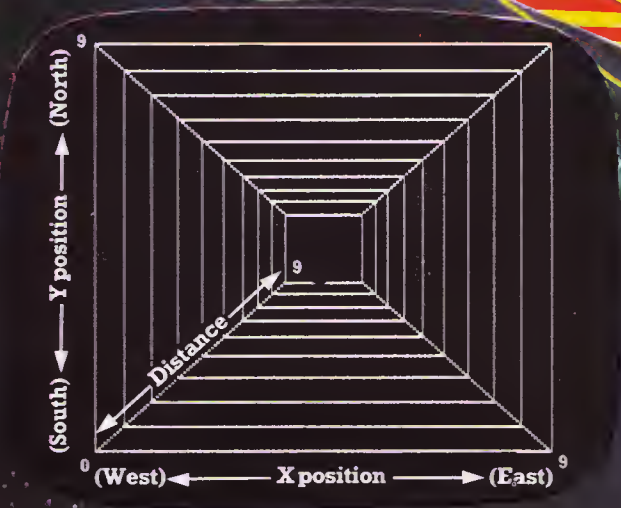
Evil Alien

Somewhere beneath you, in deepest, blackest space, lurks Elron, the Evil Alien. You have managed to deactivate all but his short-range weapons but he can still make his ship invisible. You know he is somewhere within the three-dimensional grid you can see on your ship's screen (see below), but where?

You have four space bombs. Each one can be exploded at a position in the grid specified by three numbers between 0 and 9, which your computer will ask you for. Can you blast the Evil Elron out of space before he creeps up and captures you?

```
▲●5  CLS
   10  PRINT "EVIL ALIEN"
   20  LET S=10
   30  LET G=4
★■▲●40  LET X=INT(RND*S)
★■▲●50  LET Y=INT(RND*S)
★■▲●60  LET D=INT(RND*S)

   70  FOR I=1 TO G

   80  PRINT "X POSITION (0 TO 9)?"
   85  INPUT X1
   90  PRINT "Y POSITION (0 TO 9)?"
  100  INPUT Y1
  110  PRINT "DISTANCE (0 TO 9)?"
  120  INPUT D1

  130  IF X=X1 AND Y=Y1 AND D=D1 THEN GOTO 300

  140  PRINT "SHOT WAS ";
  150  IF Y1>Y THEN PRINT "NORTH";
  160  IF Y1<Y THEN PRINT "SOUTH";
  170  IF X1>X THEN PRINT "EAST";
  180  IF X1<X THEN PRINT "WEST";
  190  PRINT
  200  IF D1>D THEN PRINT "TOO FAR"
  210  IF D1<D THEN PRINT "NOT FAR ENOUGH"

  220  NEXT I

  230  PRINT "YOUR TIME HAS RUN OUT!!"
  240  STOP
  300  PRINT "*BOOM* YOU GOT HIM!"
  310  STOP
```

Sets the size of the grid.

Sets the number of goes.

Elron's position is fixed by these 3 lines, which select 3 numbers between 0 and the size of the grid.

Start of a loop which tells the computer to repeat the next 15 lines G times.

This section asks you for your 3 numbers and stores them in X1, Y1 and D1.

Checks if you were right and jumps to 300 if you were.

Your guesses are compared with Elron's position and a report printed.

End of loop. Returns for another go.

Prints if you've used up all your goes.

Prints if you guessed right.

The above listing will work on a ZX81. For other computers, make the changes below.

```
   ●5  HOME
   ▲5  PRINT CHR$(147)
★▲●40  LET X=INT(RND(1)*S)
   ■40  LET X=INT(RND(0)*S)
★▲●50  LET Y=INT(RND(1)*S)
   ■50  LET Y=INT(RND(0)*S)
★▲●60  LET D=INT(RND(1)*S)
   ■60  LET D=INT(RND(0)*S)
```

How to make the game harder

This program has been written so that you can easily change the difficulty by changing the size of the grid. To do this, put a different value for S in line 20.

If you increase the grid size you will need more space bombs to give you a fair chance of blasting Elron. Do this by changing the value of G in line 30.

Puzzle corner

Can you work out how to change the program so that the computer asks you for a difficulty number which it can put into S instead of S being fixed? (Tip: limit the value of S to between 6 and 30 and use INT(S/3) for the value of G in line 30.)

Beat the Bug Eyes

You're trapped! Everywhere you turn you catch a glimpse of the steely cold light of a space bug's eyes before it slithers down behind a rock again. Slowly the bugs edge towards you, hemming you in, waiting for a chance to bind you in their sticky web-like extrusions. Luckily you have your proton blaster with you.

The bug eyes pop up in four different places on your screen and these correspond to keys 1 to 4. Press the correct key while the bug's eyes are on the screen and you will blast it. There are 10 bugs in all – the more you blast, the greater your chance of escape.

How the program works

```
      10 PRINT "BUG EYES"
      20 LET S=0
      30 FOR T=1 TO 10
 ▲●40 CLS
★■▲●50 FOR I=1 TO INT(RND*30+20)
      60 NEXT I
★■▲●70 LET R=INT(RND*4+1)
★■▲●80 GOSUB 210+30*R
```

Sets the score to zero for start of game.

Beginning of loop which gives you 10 turns.

Clears screen.

Delay loop which lasts a varying length of time, depending on the value of RND.
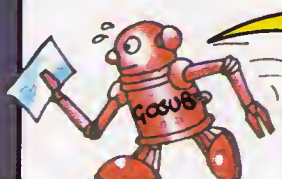
Chooses a number from 1 to 4 and puts it in R.

Jumps to one of four sub-routines depending on the value of R. This gets two numbers which correspond to a position on the screen – "A" spaces across and "D" lines down – and then jumps again to 350 to move the cursor to this position.

> GOSUB makes the computer branch out of the main program to a "sub-routine" (see next page). RETURN at the end of the sub-routine sends it back to the main program again.
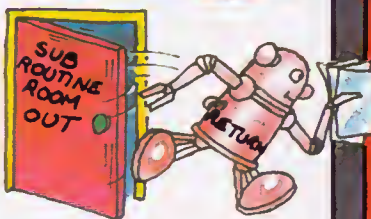
```
      90 PRINT "OO"
★■▲●100 FOR I=1 TO 20
  ★▲●110 LET R$=INKEY$
     120 IF R$<>"" THEN GOTO 140
     130 NEXT I
     140 IF VAL("O"+R$)<>R THEN GOTO 210
     150 LET S=S+1
```

Prints the "bug eyes" at this position.

Loops round to see if you are pressing a key. If you are, computer jumps to 140 and checks if it is the right one.

Increases score by 1.

106

```
▲●160  CLS  ───────────────────────────  Clears screen
   170  GOSUB 350 ⎤                        Sends cursor back to the
   180  PRINT "*" ⎦                        same position and prints a
                                           star.
★■▲●190  FOR J=1 TO 40 ⎤                   Delay loop to make star stay
   200  NEXT J        ⎦                    on screen long enough for
                                           you to see it.
   210  NEXT T ───────────────────────── Goes back for next turn.
   220  PRINT "YOU BLASTED ";S;"/10 BUGS" ── Prints score.
   230  STOP
   240  LET D=5
   250  LET A=1
   260  GOTO 350
   270  LET D=1
   280  LET A=9
   290  GOTO 350                            Sub-routines.
   300  LET D=5
   310  LET A=18
   320  GOTO 350
   330  LET D=10
   340  LET A=7
   350  FOR I=1 TO D
   360  PRINT
   370  NEXT I
   380  PRINT TAB(A);
   390  RETURN
```

The above listing will work on a ZX81. For
other computers, make the changes below.

```
    ●40,160  HOME
    ▲40,160  PRINT CHR$(147)
    ■50  FOR I=1 TO INT(RND(0)*300+200)
  ★▲●50  FOR I=1 TO INT(RND(1)*300+200)
    ■70  LET R=INT(RND(0)*4+1)
  ★▲●70  LET R=INT(RND(1)*4+1)
★■▲●80  ON R GOSUB 240,270,300,330
★■▲●100  FOR I=1 TO 150
    ●105  R$=""
    ▲110  GET R$
    ★110  R$=INKEY$(1)
    ●110  IF PEEK(-16384)>127 THEN GET R$
★■▲●190  FOR J=1 TO 300
```

How to change the speed

You can speed the game up by changing the last number in line 100 to a lower one.

How to use more of the screen

This program was written to fit on the smallest screen width (which is the VIC 20). For the other computers, you can increase the values of A in lines 250, 280, 310 and 340. Check your manual to find the maximum width your computer can use.

Puzzle corner

Can you change the program to make the bugs appear in more than four places on the screen? Can you add more bugs too?

Moonlander

You are at the controls of a lunar module which is taking a small team of astronauts down to the moon's surface. In order to land safely you must slow down your descent but that takes fuel and you have only a limited supply.

Your computer will tell you your starting height, speed and fuel supply and ask how much fuel you wish to burn. It will then work out your new height and speed. A burn of 5 will keep your speed constant. A higher number will reduce it. Try to have your speed as close to zero as you can when you land. Can *you* land safely on the moon?

> Notice the commas and semi-colons in lines 70 and 80. Experiment with leaving them out and changing them round to see what happens.

```
▲●10 CLS
   20 PRINT "MOONLANDER"
   30 LET T=0
   40 LET H=500
   50 LET V=50
   60 LET F=120
   70 PRINT "TIME";T,"HEIGHT";H
   80 PRINT "VEL.";V,"FUEL";F

   90 IF F=0 THEN GOTO 140

  100 PRINT "BURN? (0-30)"
  110 INPUT B
  120 IF B<0 THEN LET B=0
  130 IF B>30 THEN LET B=30
  140 IF B>F THEN LET B=F

  150 LET V1=V-B+5
  160 LET F=F-B
  170 IF (V1+V)/2>=H THEN GOTO 220

  180 LET H=H-(V1+V)/2
  190 LET T=T+1
  200 LET V=V1

  210 GOTO 70
  220 LET V1=V+(5-B)*H/V
  230 IF V1>5 THEN PRINT "YOU CRASHED-ALL DEAD"
  240 IF V1>1 AND V1<=5 THEN PRINT "OK-BUT SOME INJURIES"
  250 IF V1<=1 THEN PRINT "GOOD LANDING."
  260 STOP
```

The above listing will work on a ZX81. For other computers, make the changes below.

```
●10 HOME
▲10 PRINT CHR$(147)
```
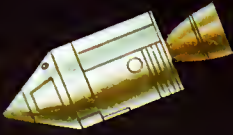
How the program works

Sets the starting values for time, height, speed and fuel and prints them.

If you have no fuel left, computer jumps down the program, bypassing the section which asks you for a burn. It then prints a running commentary of your progress as you approach the moon's surface.

Gets a number from you for the amount of fuel you wish to burn and checks it is within the correct limits.

Calculates your new speed, V1.

Calculates your new fuel level.

Checks if the distance travelled in your last go is greater or equal to your height above the moon. If it is, you've landed. Computer then jumps down program to see how good a landing you made.

Calculates your new height.

Increases time by 1.

Puts your new velocity into V so it will print in line 80 for your next go.

Goes back to beginning of loop for next go.

Calculates your speed on touch-down and checks what kind of landing it gives you.

Monsters of Galacticon

Landing on Galacticon was easy –
but no-one warned you that some of
the nastiest monsters in the known
Universe are to be found there.

As each monster threatens, you
must choose one of your weapons –
a ray gun, a trypton blaster or a
sword laser – to use against it. Can
you make the right choices? If so,
you may live to conquer Galacticon.

How the program works

```
10  PRINT "MONSTERS OF GALACTICON"

20  DIM M$(4)

30  LET N=4

40  LET M=5
50  LET M$(1)="SULFACIDOR"
60  LET M$(2)="FLAMGONDAR"
70  LET M$(3)="BALNOLOTIN"
80  LET M$(4)="GOLANDOR"
90  FOR I=1 TO N
★■▲●100  LET A=INT(RND*N+1)
★■▲●110  LET B=INT(RND*N+1)
120  LET T$=M$(A)
130  LET M$(A)=M$(B)
140  LET M$(B)=T$
150  NEXT I

160  FOR T=1 TO 8
▲●170  CLS
★■▲●180  LET R=INT(RND*N+1)
190  PRINT "MONSTER COMING..."
200  PRINT "IT'S A ";M$(R)
210  PRINT "WHICH WEAPON ? (R,S OR T) "
220  INPUT R$
s★■▲●230  LET W=CODE(R$)-54+R
★■▲240  LET W=W-3*(W>3)-3*(W>6)
250  IF W=2 THEN GOTO 300
```

Sets up a storage place (an "array") labelled M$ with 4 compartments in it – M$(1), M$(2), M$(3) and M$(4) – one for each monster name.

Sets the number of monsters to 4.

Sets the number of people in your group to 5.

Puts the 4 monster names into the array.

These lines shuffle the monsters up. Computer loops round N times. Each loop, it selects two numbers between 1 and N and switches the names in the compartments with those numbers. T$ is a temporary string used during the switch round process.
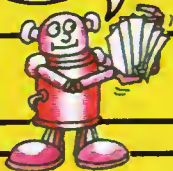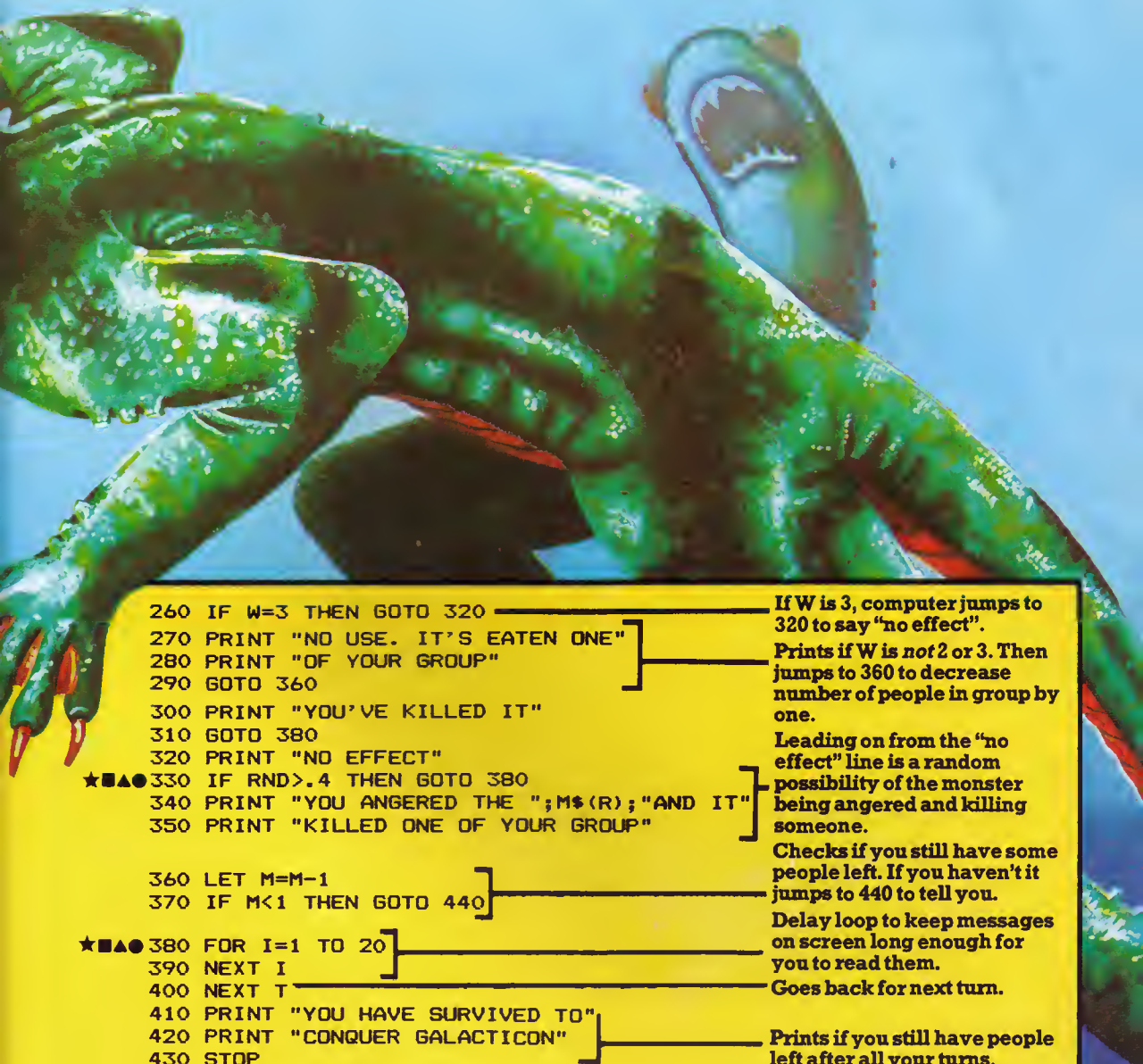
Beginning of loop for 8 turns.

Clears screen

Chooses one of the monsters and prints its name.

The computer uses the values in R and R$ to work out a weapon number, W, which will be 1, 2 or 3.

If W is 2, the computer jumps to 300 to say you've killed the monster.

You can use this shuffling routine in any program where you want things to be mixed up.

110

```
260 IF W=3 THEN GOTO 320
270 PRINT "NO USE. IT'S EATEN ONE"
280 PRINT "OF YOUR GROUP"
290 GOTO 360
300 PRINT "YOU'VE KILLED IT"
310 GOTO 380
320 PRINT "NO EFFECT"
★■▲● 330 IF RND>.4 THEN GOTO 380
340 PRINT "YOU ANGERED THE ";M$(R);"AND IT"
350 PRINT "KILLED ONE OF YOUR GROUP"

360 LET M=M-1
370 IF M<1 THEN GOTO 440
★■▲● 380 FOR I=1 TO 20
390 NEXT I
400 NEXT T
410 PRINT "YOU HAVE SURVIVED TO"
420 PRINT "CONQUER GALACTICON"
430 STOP
440 PRINT "YOU'RE ALL DEAD"
450 STOP
```

If W is 3, computer jumps to 320 to say "no effect".

Prints if W is *not* 2 or 3. Then jumps to 360 to decrease number of people in group by one.

Leading on from the "no effect" line is a random possibility of the monster being angered and killing someone.

Checks if you still have some people left. If you haven't it jumps to 440 to tell you.

Delay loop to keep messages on screen long enough for you to read them.

Goes back for next turn.

Prints if you still have people left after all your turns.

The above listing will work on a ZX81. For other computers, make the changes below.

```
    ■100 LET A=INT(RND(0)*N+1)
 ★▲●100 LET A=INT(RND(1)*N+1)
    ■110 LET B=INT(RND(0)*N+1)
 ★▲●110 LET B=INT(RND(1)*N+1)
     ●170 HOME
     ▲170 PRINT CHR$(147)
    ■180 LET R=INT(RND(0)*N+1)
 ★▲●180 LET R=INT(RND(1)*N+1)
 ★■▲●230 LET W=ASC(R$)-81+R
     $230 LET W=CODE(R$)-81+R
  ★■▲240 LET W=W+3*(W>3)+3*(W>6)
 ★▲●330 IF RND(1)>.4 THEN GOTO 380
```

```
    ■330 IF RND(0)>.4 THEN GOTO 380
 ★■▲●380 FOR I=1 TO 300
```
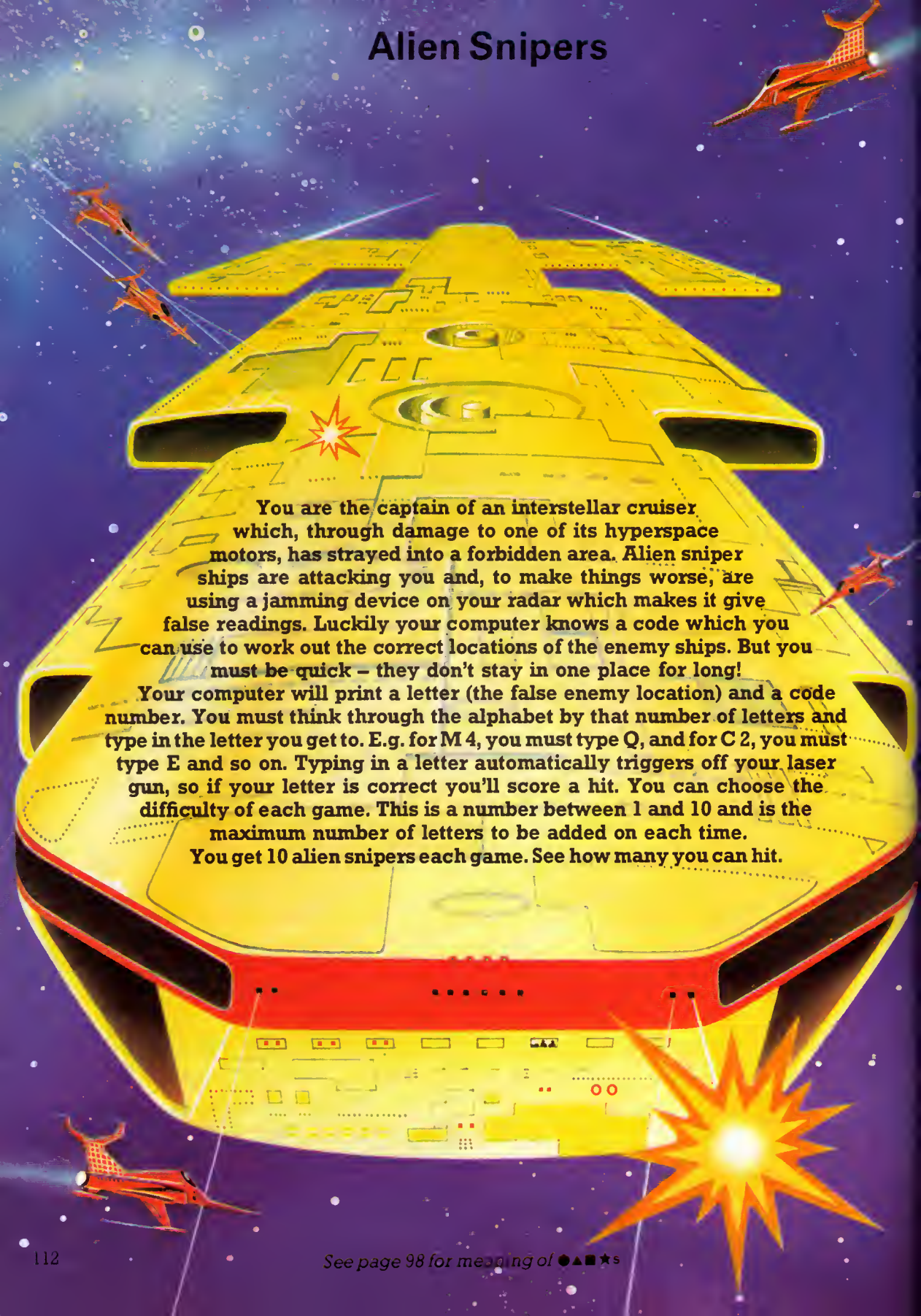
Puzzle corner

There are at least four ways of making this game harder.
Can you work out what they are?

111

TB

Alien Snipers

You are the captain of an interstellar cruiser which, through damage to one of its hyperspace motors, has strayed into a forbidden area. Alien sniper ships are attacking you and, to make things worse, are using a jamming device on your radar which makes it give false readings. Luckily your computer knows a code which you can use to work out the correct locations of the enemy ships. But you must be quick – they don't stay in one place for long!

Your computer will print a letter (the false enemy location) and a code number. You must think through the alphabet by that number of letters and type in the letter you get to. E.g. for M 4, you must type Q, and for C 2, you must type E and so on. Typing in a letter automatically triggers off your laser gun, so if your letter is correct you'll score a hit. You can choose the difficulty of each game. This is a number between 1 and 10 and is the maximum number of letters to be added on each time.

You get 10 alien snipers each game. See how many you can hit.

See page 98 for meaning of ●▲■★s

```
▲●10 CLS
   20 PRINT "ALIEN SNIPERS"
   30 PRINT
   40 PRINT "DIFFICULTY (1-10)"
   50 INPUT D
   60 IF D<1 OR D>10 THEN GOTO 50
   70 LET S=0
   80 FOR G=1 TO 10
s★■▲●90 LET L$=CHR$(INT(RND*(26-D)+38))
★■▲●100 LET N=INT(RND*D+1)
  110 CLS
  120 PRINT
  130 PRINT L$,N
★■▲●140 FOR I=1 TO 20+D*5
 ★▲●150 LET I$=INKEY$
  160 IF I$<>"" THEN GOTO 190
  170 NEXT I
  180 GOTO 200
★■▲●190 IF I$=CHR$(CODE(L$)+N) THEN LET S=S+1
  200 NEXT G
  210 PRINT "YOU HIT ";S;"/10"
  220 STOP
```

1K

Gets a difficulty number from you, puts it in D, and checks it is within the correct limits.

Sets opening score to zero.

Beginning of loop which gives you 10 goes.

Selects a letter between A and the letter which is your difficulty number before the end of the alphabet.

Selects a number between 1 and D.

Prints the letter and the number.

Checks if you are pressing a key and jumps to 190 if you are.

If you didn't press a key, computer jumps to line 200 which sends it back for another go.
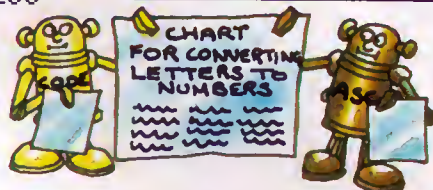
Checks if you the pressed right key and, if so, increases your score by 1.

End of loop. Goes back for another turn.

Prints score after 10 goes.

The above listing will work on a ZX81. For other computers, make the changes below.

```
    ●10,110 HOME
    ▲10,110 PRINT CHR$(147)
    ■90 LET L$=CHR$(INT(RND(0)*(26-D)+65))
 ★▲●90 LET L$=CHR$(INT(RND(1)*(26-D)+65))
    s90 LET L$=CHR$(INT(RND *(26-D)+65))
    ■100 LET N=INT(RND(0)*D+1)
 ★▲●100 LET N=INT(RND(1)*D+1)
 ★■▲140 FOR I=1 TO 200+D*50
    ●140 FOR I=1 TO 100+D*50
    ●145 I$=""
    ★150 LET I$=INKEY$(1)
    ▲150 GET I$
    ●150 IF PEEK(-16384)>127 THEN GET I$
★■▲●190 IF I$=CHR$(ASC(L$)+N) THEN LET S=S+1
```

How to change the speed

If the game is too quick for you, put a higher number into the middle of line 140 (i.e. to replace 20 or 200). You can speed it up with a lower number.

How to make the game harder

You could change the 1 in lines 40 and 60 to, say, 3 to allow difficulties of only 3 or more.

Puzzle corner

Can you adjust the scoring so that it fits the code number i.e. you score 1 point if the code is 1, 2 points if the code is 2 and so on?

Asteroid Belt

You are on a trip through the Asteroid Belt. To avoid crashing into the asteroids, you must destroy them and the force required for doing this depends on their size.

Asteroids appear on your computer screen as groups of stars. To destroy them you must press the number key corresponding to the number of stars. Be prepared – asteroids come at you thick and fast.

How the program works

```
10  PRINT "ASTEROID BELT"

20  LET S=0

30  FOR G=1 TO 10

▲●40  CLS

★■▲●50  LET A=INT(RND*18+1)

★■▲●60  LET D=INT(RND*12+1)

★■▲●70  LET N=INT(RND*9+1)

80  FOR I=1 TO D
90  PRINT
100  NEXT I

110  FOR I=1 TO N
120  IF I<>1 AND I<>4 AND I<>7 THEN GOTO 150
130  PRINT
140  PRINT TAB(A);
150  PRINT "*";
160  NEXT I

170  PRINT

★■▲●180  FOR I=1 TO 10
★▲●190  LET Q=VAL("0"+INKEY$)
200  IF Q<>0 THEN GOTO 240
★210  NEXT I

220  PRINT "CRASHED INTO ASTEROID"
230  GOTO 290

240  IF Q<>N THEN GOTO 270
```

Sets the opening score to zero.

Starts a loop which gives 10 goes.

Chooses a number for the position of the asteroid across the screen. Puts this in A.

Chooses a number (1 to 12) for the position of the asteroid down the screen and puts it in D.

Chooses a number (1 to 9) for the number of stars in the asteroid.

Moves the cursor D lines down the screen.

Loops round N times printing a star each time in the appropriate position.

Loops round to see if you are pressing a key and jumps to 240 if you are.

Prints if you run out of time.

Checks if your number is the same as N and jumps to 270 if it isn't.

114

```
        250  PRINT "YOU DESTROYED IT"————————————— Prints if you pressed the right
                                                              number.

        260  LET S=S+1 ———————————————————————————— Increases your score by one.

        270  IF Q<N THEN PRINT "NOT STRONG ENOUGH"     Compares your number with
        280  IF Q>N THEN PRINT "TOO STRONG"            N and prints an appropriate
                                                              message.

★■▲●290  FOR I=1 TO 50                                Delay loop to keep messages
        300  NEXT I                                   on screen long enough for
                                                              you to read them.

        310  NEXT G ——————————————————————————————— Goes back for another go.

        320  PRINT "YOU HIT ";S;" OUT OF 10"          Prints your score after 10
        330  STOP                                     goes.
```

The above listing will work on a ZX81. For other computers, make the changes below.

```
    ●40   HOME
    ▲40   PRINT CHR$(147)
    ■50   LET A=INT(RND(0)*18+1)
★▲●50   LET A=INT(RND(1)*18+1)
    ■60   LET D=INT(RND(0)*12+1)
★▲●60   LET D=INT(RND(1)*12+1)
    ■70   LET N=INT(RND(0)*9+1)
★▲●70   LET N=INT(RND(1)*9+1)
    ●175  Q=0
    ★180
■▲●180  FOR I=1 TO 100
    ▲190  GET Q
    ●190  IF PEEK(-16384)>127 THEN GET Q
    ★190  Q=INKEY(100)-48
    ★210
★■290  FOR I=1 TO 500
▲●290  FOR I=1 TO 250
```

Changing the speed of the game

Line 180 (190 for the BBC) controls how much time you have to press a key. Change the last number in 180, or the number in brackets in the BBC line 190, to a lower number to speed up the game.

Puzzle corner

Can you adapt the scoring system so that, for each asteroid, you get the same number of points as there are stars in it?

Trip into the Future

Imagine you are in a spaceship travelling nearly as fast as light. Strangely time is passing more slowly inside your spaceship than outside. So, having set off on a long, fast space trip, you can return to Earth further in the future than the clocks inside your ship indicate.

In this game, your computer tells you how many years must elapse on Earth before you return. You then decide the length of your trip (in light years) and the speed of your ship (as a fraction of the speed of light) in order to achieve this. Take care not to travel too far too slowly or you will die of old age on the way.

How the program works

```
▲●10  CLS
   20  PRINT "TRIP INTO THE FUTURE"
★■▲●30  LET T=INT(RND*100+25)
   40  PRINT "YOU WISH TO RETURN ";T
   50  PRINT "YEARS IN THE FUTURE."
   60  PRINT
   70  PRINT "SPEED OF SHIP (0-1)"
   80  INPUT V
   90  IF V>=1 OR V<=0 THEN GOTO 70
  100  PRINT "DISTANCE OF TRIP"
  110  INPUT D
  120  LET T1=D/V
```

Chooses a whole number between 25 and 124 for the years which must elapse before your return, and prints it.

Gets a speed from you and checks it is within the correct limits.

Gets a distance from you.

Calculates the time taken *inside* the ship.

See page 98 for meaning of ●▲■★
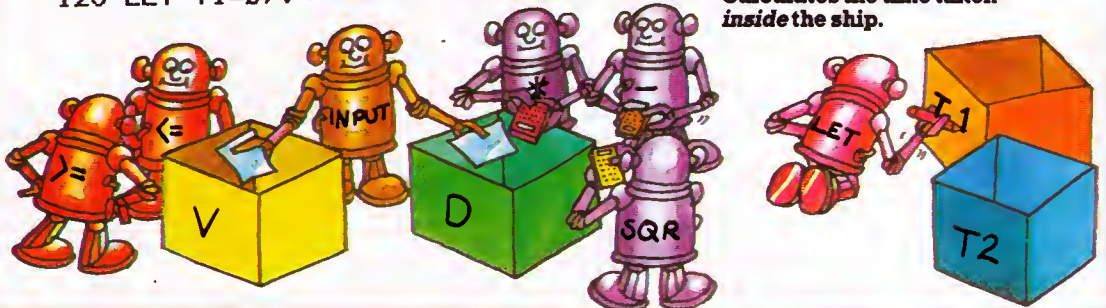
```
130 LET T2=T1/SQR(1-V*V)
```
Calculates the time taken *outside* the ship (i.e. on Earth).

```
140 PRINT "YOU TOOK ";T1;"YEARS"
150 PRINT "AND ARRIVED ";T2;"YEARS"
160 PRINT "IN THE FUTURE."
```
Prints these times.

```
170 IF T1>50 THEN GOTO 210
```
Checks if you took longer than your lifetime (50 years). Jumps to line 210 if you did.

```
180 IF ABS(T-T2)<=5 THEN PRINT "YOU ARRIVED ON TIME"
190 IF ABS(T-T2)>5 THEN PRINT "NOT EVEN CLOSE"
200 STOP
210 PRINT "YOU DIED ON THE WAY"
220 STOP
```
Checks if you were within 5 years and prints a message.

The above listing will work on a ZX81. For other computers, make the changes below.

```
●10  HOME
▲10  PRINT CHR$(147)
■30  LET T=INT(RND(0)*100+25)
★▲●30 LET T=INT(RND(1)*100+25)
```

Puzzle corner
Can you work out how to change the program to do the following things?

1) Give a wider range of years which must elapse before you return to Earth.
2) Increase the accuracy required from within 5 years to within 2 years.
3) Shorten or lengthen your lifetime.

117

Death Valley

There is only one way to escape the forces of the evil Dissectitrons. You will have to steel every nerve and fly your single-seater Speed Dart along the jagged, bottomless ravine known as Death Valley.

Your computer will ask you for the width of the valley. Try 15* first and then work your way down – 8 is quite difficult. Steer your Speed Dart by pressing Q to go left and P to go right, and see if you can make it safely through Death Valley.

*If you are using a VIC 20, then use widths of 6 to 10.

How the program works

```
10 PRINT "DEATH VALLEY"
20 LET S=0
```
Sets the number of goes used to zero for start of game.

```
30 LET M=200
```
M is the maximum number of goes allowed.

```
40 PRINT "WIDTH?"
50 INPUT W
60 LET W=INT(W/2)
```
Gets a width number from you, divides it by 2 and uses INT to remove any halves.

```
▲70 LET L=10
80 LET Y=W
90 LET R=W
```
L, Y and R are the distances between walls and Speed Dart.

Edge of screen | L spaces (= 10 at start) | I | Y spaces (= W at start) | * | R spaces (= W at start) | I

```
★■▲●100 LET D=INT(RND*3-1)
```
Selects – 1, 0 or +1 and puts it in D.
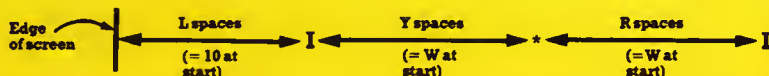
```
110 IF L+D<0 OR L+D>20 THEN GOTO 100
```
Checks L + D isn't so big or small that columns disappear off sides of screen.

```
120 LET L=L+D
130 LET Y=Y-D
140 LET R=R+D
```
Changes spaces between columns according to the value of D.

← +D → I ← −D → * ← +D → I

```
S★■▲●145 SCROLL
150 LET N=L
160 GOSUB 1000
170 PRINT "I";
```
Moves cursor L spaces across the screen and prints I. (The semi-colon then stops the cursor going down to the next line.)

```
180 LET N=Y
190 GOSUB 1000
200 PRINT "*";
```
Moves cursor a further Y spaces and prints.*

```
210 LET N=R
220 GOSUB 1000
230 PRINT "I"
```
Moves cursor a further R spaces and prints another I. (No semi-colon this time, so the cursor then goes down to the next line.)

```
★▲●240 LET I$=INKEY$
```
Checks to see if you are pressing a key.

```
250 IF I$<>"Q" THEN GOTO 280
```
If the key you are pressing is not Q, computer jumps to 280.

```
260 LET Y=Y-1
270 LET R=R+1
```
If key is Q then Y is decreased by one and R increased by one so star moves left.

```
280 IF I$<>"P" THEN GOTO 310
```
Checks if key being pressed is P.

18

```
      290 LET Y=Y+1
      300 LET R=R-1
```

If so, star is moved right by increasing Y by one and decreasing R by one.

```
      310 IF Y<1 OR R<1 THEN GOTO 370
```

Checks if you have crashed into a wall. Jumps to 370 to tell you if you have.

```
      320 LET S=S+1
```

Increases number of goes used by one.

```
      330 IF S<M THEN GOTO 100
```

Goes back for another turn if you have had less than M goes.

```
      340 PRINT "WELL DONE-YOU MADE IT".
      350 PRINT "THROUGH DEATH VALLEY"
```

Prints if you have had M goes and no crash.

```
      360 STOP
      370 PRINT "YOU CRASHED INTO THE WALL"
      380 PRINT "AND DISINTEGRATED"
      390 STOP
     1000 IF N=0 THEN RETURN
     1010 FOR I=1 TO N
     1020 PRINT " ";
     1030 NEXT I
     1040 RETURN
```

Sub-routine for moving the cursor to the appropriate places for printing I and *.

The above listing will work on a ZX81. For other computers, make the changes below.

```
    ▲70 LET L=4
    ■100 LET D=INT(RND(0)*3-1)
 ★▲●100 LET D=INT(RND(1)*3-1)
s★■▲●145
    ●235 I$=""
    ▲240 GET I$
    ★240 LET I$=INKEY$(1)
    ●240 IF PEEK(-16384)>127 THEN GET I$
```

Slowing down the game

If this game is too fast for you, you can add a delay loop at lines 141 and 142:
```
141 FOR J=1 TO 100
142 NEXT J
```
Change the number in line 141 to adjust the speed – the lower the number the faster the game.

Puzzle corner

How can you make the valley longer?

119

Space Mines

You are the newly elected leader of a mining colony on the Planet Astron. All decisions concerning the sale of ore to Intergalactic Traders, food purchase and sale and purchase of mines are made by you. Can you keep the people satisfied and survive your 10 years in office or will life in the colony end in disaster under your rule?

How the program works

```
★■▲● { 10 LET L=INT(RND*3+5)
       20 LET P=INT(RND*60+40)
       30 LET M=INT(RND*50+10)*P
       40 LET FP=INT(RND*40+80)
       50 LET CE=INT(RND*40+80)
```

These lines decide the number of mines (L), number of people (P), amount of money (M), price of food (FP) and amount of ore produced per mine (CE) for the start of the game.

```
       60 LET C=0
```
Sets the amount of ore in storage to zero for start.

```
       70 LET S=1
```
Sets the satisfaction factor to 1.

```
       80 LET Y=1
```
Sets the year number to 1.

```
★■▲● 90 LET LP=INT(RND*2000+2000)
```
Selects buying/selling price for mines.

```
★■▲● 100 LET CP=INT(RND*12+7)
```
Selects selling price for ore.

```
  ▲● 110 CLS
      120 PRINT "YEAR";Y
      130 PRINT
      140 PRINT "THERE ARE ";P;" PEOPLE IN THE COLONY."
      150 PRINT "YOU HAVE ";L;" MINES,AND $"M
      160 PRINT "SATISFACTION FACTOR ";S
      170 PRINT
      180 PRINT "YOUR MINES PRODUCED ";CE;" TONS EACH"
      190 LET C=C+CE*L
      200 PRINT "ORE IN STORE=";C;" TONS"
      210 PRINT "SELLING"
      220 PRINT "ORE SELLING PRICE=";CP
      230 PRINT "MINE SELLING PRICE=";LP;"/MINE"
```
Prints current state of affairs in the colony.

```
      240 PRINT "HOW MUCH ORE TO SELL?"
      250 INPUT CS
      260 IF CS<0 OR CS>C THEN GOTO 240
```
Asks how much ore you want to sell, puts it CS and checks you've got that much in store.

```
      270 LET C=C-CS
```
Takes amount sold away from amount in store.

```
      280 LET M=M+CS*CP
```
Works out how much ore sold is worth and adds this to your money supply.

```
      290 PRINT "HOW MANY MINES TO SELL?"
      300 INPUT LS
      310 IF LS<0 OR LS>L THEN GOTO 290
      320 LET L=L-LS
      330 LET M=M+LS*LP
```
Does the same process for selling mines.

```
      340 PRINT
      350 PRINT "YOU HAVE $";M
      360 PRINT
```
Prints your new money supply.

```
      370 PRINT "BUYING"
      380 PRINT "HOW MUCH TO SPEND ON FOOD ? (APPR. $100 EA.) "
      390 INPUT FB
```
Asks how much you want to spend on food and puts this in FB.

```
      400 IF FB<0 OR FB>M THEN GOTO 380
```
Checks you have enough money to pay.

```
      410 LET M=M-FB
```
Adjusts your money supply.

120

```
420 IF FB/P>120 THEN LET S=S+.1
430 IF FB/P<80 THEN LET S=S-.2
```
Adjusts satisfaction factor according to how much you spent on food.

```
440 PRINT "HOW MANY MORE MINES TO BUY?"
450 INPUT LB
460 IF LB<0 OR LB*LP>M THEN GOTO 440
```
Asks how many mines you want to buy and checks you can afford them.

```
470 LET L=L+LB
```
Increases number of mines if necessary.

```
480 LET M=M-LB*LP
```
Adjusts your money again.

```
490 IF S<.6 THEN GOTO 660
```
Checks on value of satisfaction factor. If this is very low, computer jumps to 660 to end the game.

```
★■▲● 500 IF S>1.1 THEN LET CE=CE+INT(RND*20+1)
```
If S is high then amount produced per mine is increased.

```
★■▲● 510 IF S<.9 THEN LET CE=CE-INT(RND*20+1)
```
If S low, amount produced is decreased.

```
520 IF P/L<10 THEN GOTO 680
```
If there are less than 10 people per mine, game is over.

```
★■▲● 530 IF S>1.1 THEN LET P=P+INT(RND*10+1)
```
More people arrive if S is high.

```
★■▲● 540 IF S<.9 THEN LET P=P-INT(RND*10+1)
```
People leave if S is low.

```
550 IF P<30 THEN GOTO 700
```
If there are less than 30 people, game is over.

```
★■▲● 560 IF RND>.01 THEN GOTO 590
570 PRINT "RADIOACTIVE LEAK....MANY DIE"
580 LET P=INT(P/2)
```
Introduces small chance of half the people being killed.

```
590 IF CE<150 THEN GOTO 620
600 PRINT "MARKET GLUT - PRICE DROPS"
610 LET CE=INT(CE/2)
```
If the amount produced per mine is very high, ore price is halved.

```
620 LET Y=Y+1
```
Year number increased by 1 and if it is less than 11, computer goes back to 90 for another go.

```
630 IF Y<11 THEN GOTO 90
640 PRINT "YOU SURVIVED YOUR TERM OF OFFICE"
650 STOP
660 PRINT "THE PEOPLE REVOLTED"
670 STOP
680 PRINT "YOU'VE OVERWORKED EVERYONE"
690 STOP
700 PRINT "NOT ENOUGH PEOPLE LEFT"
710 STOP
```
Prints if the computer reaches this stage in the program on your 10th go.

The above listing will work on a ZX81. For other computers, make the changes below.

```
● 110 HOME
▲ 110 PRINT CHR$(147)
★▲● 10,20,30,40,50,90,100,500,510,530,540,560
    change RND to RND(1)
■ 10,20,30,40,50,90,100,500,510,530,540,560
    change RND to RND(0)
```

See page 98 for meaning of ●▲■★

121

Space Rescue

You must make an urgent trip across the spiral arm of the Galaxy to a developing planet which is in need of medical supplies. The trip involves such huge distances that for most of it you will be in a deep sleep, but before this you must program the ship for the journey. The computer will ask how much energy you want to allocate to the engines, life support system and shields and then put you to sleep.

When you wake up, it will give you a report on what happened during the trip and, if all went well, you will be orbiting the planet. You must now allocate your remaining energy to the landing boosters and shields in order to make a good landing on the planet.

If you accomplish the mission safely, you stand a good chance of being promoted to Space Admiral. Good luck!

```
▲●10 CLS
    20 PRINT "SPACE RESCUE"
    30 PRINT
    40 PRINT "DO YOU WANT INSTRUCTIONS? "
    50 INPUT I$
★■▲●60 IF I$(1)="Y" THEN GOSUB 1000
★■▲●70 LET D=INT(RND*800+101)
★■▲●80 LET E=INT(RND*400+401)
    90 LET T=INT(D/SQR(E/5)+.5)
    100 PRINT "THE PLANET IS ";D;" UNITS AWAY"
    110 PRINT "YOU HAVE ";E;" UNITS OF ENERGY"
    120 PRINT "AND A TIME LIMIT OF ";T;" DAYS"
    130 PRINT
    140 PRINT "ENERGY DISTRIBUTION:"
    150 PRINT "TO ENGINES?"
    160 INPUT P
    170 PRINT "TO LIFE SUPPORT?"
    180 INPUT L
    190 PRINT "TO SHIELDS?"
    200 INPUT S
    210 IF P+L+S>E THEN GOTO 140
    220 LET X=E-P-L-S
    230 LET V=INT(SQR(P))
    240 LET T1=INT(D/V)
▲●250 CLS
    260 PRINT "YOUR VELOCITY IS ";V
    270 PRINT "YOU HAVE AN ETA OF ";T1;" DAYS"
    280 PRINT
★■▲●290 FOR I=1 TO INT(RND*5+6)
★■▲●300 IF RND>.5 THEN GOTO 430
★■▲●310 GOTO 320+INT(RND*4)*30
    320 PRINT "ASTEROID STORM - SHIELDS DAMAGED"
```

```
★■▲●330 LET S=S-20-INT(RND*40+1)
   340 GOTO 430
   350 PRINT "COMPUTER BREAKDOWN - DELAY IN REPAIRING"
★■▲●360 LET D=D+INT(RND*20+1)
   370 GOTO 430
   380 PRINT "ENGINE TROUBLE - MUST SLOW DOWN"
   390 LET V=V-.5
   400 GOTO 430
   410 PRINT "X-RAY DAMAGE - LIFE SUPPORT DAMAGED"
   420 LET L=L-20-INT(RND*40+1)
★■▲●430 FOR J=1 TO 50
   440 NEXT J
   450 NEXT I
   460 LET T1=INT(D/V)
  ▲●470 CLS
   480 PRINT "ARRIVED IN ";T1;" DAYS"
   490 IF S<0 THEN PRINT "SHIELDS DESTROYED"
   495 IF S<0 THEN PRINT "YOU WERE BLOWN UP"
   500 IF L<=0 THEN PRINT "LIFE SUPPORT INACTIVE"
   505 IF L<=0 THEN PRINT "YOU'RE DEAD"
   510 IF V<=0 THEN PRINT "ENGINES ARE NON-FUNCTIONAL"
   520 IF T1>T THEN PRINT "YOU TOOK TOO LONG ABOUT IT"
   530 IF S<0 OR L<=0 OR V<=0 OR T1>T THEN STOP
★■▲●540 LET G=INT(RND*10+5)
   550 LET G$="HIGH"
   560 IF G<12 THEN LET G$="MEDIUM"
   570 IF G<8 THEN LET G$="LOW"
★■▲●580 LET A=INT(RND*10+5)
   590 LET A$="HIGH"
   600 IF A<12 THEN LET A$="MEDIUM"
   610 IF A<8 THEN LET A$="LOW"
   620 PRINT
   630 PRINT "YOU ARE NOW ORBITING THE PLANET"
   640 PRINT "SURPLUS ENERGY=";X
   650 PRINT "GRAVITY IS ";G$
   660 PRINT "ATMOSPHERE IS ";A$
   670 PRINT
   680 PRINT "HOW MUCH ENERGY TO BOOSTERS?"
   690 INPUT B
   700 PRINT "HOW MUCH ENERGY TO HEAT SHIELDS?"
   710 INPUT S
   720 IF B+S>X THEN GOTO 680
  ▲●730 CLS
   740 IF B>=G*10 THEN GOTO 770
```

Program continued on next page.

```
750 PRINT "YOU MADE A NEW CRATER"
760 GOTO 840
770 IF S>=A*10 THEN GOTO 800
780 PRINT "YOU MADE A WONDERFUL SHOOTING STAR"
790 GOTO 840
800 PRINT "YOU LANDED SUCCESSFULLY - WELL DONE"
810 IF X-S-B>25 THEN GOTO 840
820 PRINT "PITY YOU DON'T HAVE ENOUGH"
830 PRINT "ENERGY TO OPEN THE DOOR"
840 STOP
1000 PRINT
1010 PRINT "YOU ARE ABOUT TO EMBARK ON A"
1020 PRINT "MISSION TO A DISTANT PLANET"
1030 PRINT "IN URGENT NEED OF MEDICAL"
1040 PRINT "SUPPLIES. YOU MUST FIRST READY"
1050 PRINT "YOUR SHIP FOR THE TRIP BY"
1060 PRINT "ALLOCATING SOME OF THE SHIP'S"
1070 PRINT "ENERGY TO THE ENGINES,SHIELDS"
1080 PRINT "AND LIFE-SUPPORT. YOU ARE"
1090 PRINT "THEN PUT TO SLEEP FOR THE MAIN"
1100 PRINT "PART OF THE TRIP, AFTER WHICH"
1110 PRINT "YOU WILL GET A REPORT OF THE"
1120 PRINT "EVENTS ON THE WAY. YOU MUST"
1130 PRINT "THEN LAND ON THE PLANET....."
1140 PRINT "PRESS ANY KEY"
★■▲●1150 IF INKEY$="" THEN GOTO 1150
▲●1160 CLS
1170 RETURN
```

The above listing will work on a ZX81. For
other computers, make the changes below.

```
■all RND to RND(0)
★▲●all RND to RND(1)
●10,250,470,730,1160 HOME
▲10,250,470,730,1160 PRINT CHR$(147)
★■▲●60 IF LEFT$(I$,1)="Y" THEN GOSUB 1000
★■▲●310 ON INT(RND*4+1) GOTO 320,350,380,410
▲●430 FOR J=1 TO 500
★430 FOR J=1 TO 1000
★1150 I=GET
●1150 GET I$
■▲1150 GET I$ : IF I$="" THEN GOTO 1150
```

Adding to the game

This game is really made up of two parts. In the first part you set off on your space journey with the aim of orbiting the planet and in the second you attempt to land on the planet. You could perhaps try adding a third part in which you make the treacherous crossing from the landing site to the Intergalactic Red Cross H.Q.

Touchdown

This game is different from the others in this book because it uses graphics. As the computers vary so much in the way their graphics work, there is a separate program for each one. Read the instructions on this page for how to play the game and then look through the pages that follow for the version for your computer.

How to play Touchdown

Ace space pilot, Captain Flash, is sitting next to you as you take the final part of your Advanced Spacecraft Handling Test (Part III). Your lightweight, two-man landing craft is rapidly approaching the Moon's surface. Your velocity must be almost zero as you touch down. Deftly you control the thrust, pressing A to increase it and D to decrease it*, watching your progress on the screen all the time. If you use too much thrust you will begin to go back up again. Too little and you will make a new crater on the Moon. Can you impress Captain Flash with your skill?

*For VIC, use the cursor down key to increase thrust and the cursor right key to decrease it.

Touchdown: TRS-80 version

```
20 CLS
30 CLEAR 200
31 B$=STRING$(25,131)
33 M1$=CHR$(194)+STRING$(2,176)
34 M2$=" "+STRING$(4,191)
35 M3$=CHR$(131)+CHR$(135)+STRING$
   (2,131)+CHR$(139)+CHR$(131)
40 GOSUB 250
50 GOSUB 300
60 C=1:GOSUB 390
70 A=1:B=F:GOSUB 460
80 A=2:B=ABS(V):GOSUB 460
90 A=3:B=H:GOSUB 460
100 A=4:B=T:GOSUB 460
110 GOSUB 530
120 V1=V-T/20+G : F=F-T/10
130 H1=H-(V+V1)/10
140 C=0:GOSUB 390
150 IF H1<0 THEN 200
160 H=H1:V=V1
170 IF H<=100 THEN 60
180 GOSUB 590
190 GOTO 220
200 H=0:C=1:GOSUB 390
210 GOSUB 660
220 END
250 H=100:F=100:T=0
260 V=INT(RND(0)*10+6)
270 G=INT(RND(0)*40+41)/100
280 RETURN
300 FOR X=80 TO 127
320 SET (X,47-INT(RND(0)*5))
330 NEXT
340 PRINT "GRAVITY=";G
350 PRINT @192,"FUEL:"
355 PRINT @384,"VEL:"
360 PRINT @576,"HEIGHT:"
365 PRINT @768,"THRUST:"
370 RETURN
390 Y=818-64*INT(H/8)
400 PRINT @Y,; : IF C=1 THEN PRINT
    M1$; ELSE PRINT CHR$(196);
410 PRINT @Y+64,; : IF C=1 THEN
    PRINT M2$; ELSE PRINT CHR$
    (198);
420 PRINT @Y+128,; : IF C=1 THEN
    PRINT M3$; ELSE PRINT CHR$
    (198)
440 RETURN
460 Y=(A*3+1)*64
470 PRINT @Y,CHR$(217);
480 PRINT @Y,LEFT$(B$,B/4);
510 RETURN
530 I$=INKEY$
540 IF I$="A" THEN T=T+4 : IF
    T>100 THEN T=100
550 IF I$="D" THEN T=T-4 : IF
    T<0 THEN T=0
560 IF T>F THEN T=F
570 RETURN
590 CLS
600 FOR I=1 TO 20
610 PRINT @INT(RND(0)*1024),"*"
620 NEXT
630 PRINT @470,"LOST IN SPACE!!"
640 RETURN
650 CLS
660 PRINT "LANDED AT VEL ";
    INT((V+V1)*5)/10
670 IF (V+V1)<8 THEN PRINT "SAFELY"
    ELSE PRINT "ALL DEAD"
680 RETURN
```
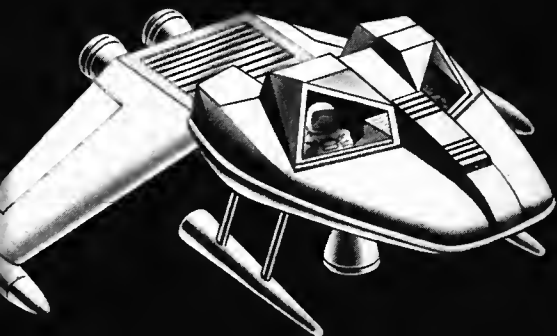
Touchdown: VIC 20 version

```
20 PRINT CHR$(147)CHR$(5);
25 POKE 36879,8
30 DEF FNR(X)=INT(RND(1)*X+1)
40 GOSUB 250
50 GOSUB 300
60 C=1:GOSUB 390
70 A=1:B=F:GOSUB 460
80 A=2:B=ABS(V):GOSUB 460
90 A=3:B=H:GOSUB 460
100 A=4:B=T:GOSUB 460
110 GOSUB 530
120 V1=V-T/20+G : F=F-T/10
130 H1=H-(V+V1)/10
140 C=0:GOSUB 390
150 IF H1<0 THEN 200
160 H=H1:V=V1
170 IF H<=100 THEN 60
180 GOSUB 590
190 GOTO 220
200 H=0:C=1:GOSUB 390
210 GOSUB 660
220 END
250 H=100:F=100:T=0
260 V=5+FNR(10)
270 G=(FNR(40)+40)/100
280 RETURN
300 FOR X=8178 TO 8185
320 POKE X,98+2*FNR(3)
330 NEXT
340 PRINT "GRAVITY=";G
350 PRINT "QQFUEL:"
355 PRINT "QQQVEL:"
360 PRINT "QQQHEIGHT:"
365 PRINT "QQQTHRUST:"
370 RETURN
```

```
390 Y=8137-22*INT(H/5)
400 IF C=0 THEN 425
405 POKE Y,108 : POKE Y+1,123
410 POKE Y+22,160 : POKE Y+23,160
415 POKE Y+44,75 : POKE Y+45,74
420 GOTO 440
425 FOR Z=0 TO 44 STEP 22
430 POKE Y+Z,32 : POKE Y+Z+1,32
435 NEXT
440 RETURN
460 FOR X=0 TO 9
470 Y=A*88+X+7724
480 IF X<B/10 THEN POKE Y,102 :
      GOTO 500
485 IF X<B/10+.5 THEN POKE Y,92 :
      GOTO 500
490 POKE Y,32
500 NEXT
510 RETURN
530 GET I$
540 IF I$="Q" THEN T=T+4 : IF T>100
      THEN T=100
550 IF I$="]" THEN T=T-4 : IF T<0
      THEN T=0
560 IF T>F THEN T=F
570 RETURN
590 PRINT CHR$(147)
600 FOR I=1 TO 20
610 POKE 7679+FNR(506),42
620 NEXT
630 PRINT "LOST IN SPACE!!"
640 RETURN
650 PRINT CHR$(147)"LANDED"
660 PRINT "AT VEL ";INT((V+V1)*5)/10
670 IF (V+V1)<8 THEN PRINT "SAFELY"
      :RETURN
680 PRINT "ALL DEAD":RETURN
```

Q is cursor down key

] is cursor right key

Touchdown: ZX81 version

```
20 CLS
30 LET B$="[6666666666666666666666]"
40 GOSUB 250
50 GOSUB 300
60 LET C=1
65 GOSUB 390
70 LET A=1
75 LET B=F
77 GOSUB 460
80 LET A=2
85 LET B=ABS(V)
87 GOSUB 460
90 LET A=3
95 LET B=H
97 GOSUB 460
100 LET A=4
105 LET B=T
107 GOSUB 460
110 GOSUB 530
120 LET V1=V-T/20+G
125 LET F=F-T/10
130 LET H1=H-(V+V1)/10
140 LET C=0
145 GOSUB 390
150 IF H1<0 THEN GOTO 200
160 LET H=H1
165 LET V=V1
170 IF H<=100 THEN GOTO 60
180 GOSUB 590
190 GOTO 220
200 LET H=0
205 LET C=1
207 GOSUB 390
210 GOSUB 660
220 STOP
250 LET H=100
252 LET F=100
255 LET T=0
260 LET V=5+INT(RND*10+1)
270 LET G=INT(RND*40+40)/100
280 RETURN
300 PRINT AT 21,20;"[SDADSADAS]"
340 PRINT AT 0,0;"GRAVITY=";G
350 PRINT AT 3,0;"FUEL:"
355 PRINT AT 7,0;"VEL:"
360 PRINT AT 11,0;"HEIGHT:"
365 PRINT AT 15,0;"THRUST:"
370 RETURN
390 LET Y=(100-H)/6+1
400 IF C=0 THEN GOTO 425
405 PRINT AT Y,24;"[34]"
410 PRINT AT Y+1,23;"[Q  W]"
415 PRINT AT Y+1,23;"[TGGY]"
420 GOTO 440
425 PRINT AT Y,24;"   "
430 PRINT AT Y+1,23;"     "
435 PRINT AT Y+2,23;"     "
440 RETURN
460 LET Y=A*4
470 PRINT AT Y,0;TAB(20)
480 PRINT AT Y,0;B$( TO B/5)
```

```
510 RETURN
530 LET I$=INKEY$
540 IF I$="A" THEN LET T=T+4
545 IF T>100 THEN LET T=100
550 IF I$="D" THEN LET T=T-4
555 IF T<0 THEN LET T=0
560 IF T>F THEN LET T=F
570 RETURN
590 CLS
600 FOR I=1 TO 20
610 PRINT AT INT(RND*22),
    INT(RND*32);"*"
620 NEXT I
630 PRINT AT 10,10;"LOST IN SPACE"
640 RETURN
650 CLS
660 PRINT AT 20,0;"LANDED AT VEL "
    ;INT((V+V1)*5)/10
670 IF (V+V1)<8 THEN PRINT "SAFELY"
675 IF (V+V1)>=8 THEN PRINT
    "ALL DEAD"
680 RETURN
```

128

Special note for ZX81 users

Graphic characters are shown in the listing using the following convention. A '[' character means press the GRAPHICS key, and then type the following letters/numbers SHIFTED until ']' is reached, when you type the GRAPHICS key again, and go back to normal.

Touchdown: ZX Spectrum version

```
20 CLS
30 DEF FNr(x)=INT(RND*x+1)
40 GOSUB 250
50 GOSUB 300
60 LET c=0: GOSUB 390
70 LET a=1: LET b=f: LET c=2*(f<25)
75 GOSUB 460
80 LET a=2: LET b=ABS v:
   LET c=4*(v<0)
85 GOSUB 460
90 LET a=3: LET b=h: LET c=2*(h<25)
95 GOSUB 460
100 LET a=4: LET b=t: LET c=0
105 GOSUB 460
110 GOSUB 530
120 LET v1=v-t/20+g: LET f=f-t/10
130 LET h1=h-(v+v1)/10
140 LET c=1: GOSUB 390
150 IF h1<0 THEN GOTO 200
160 LET h=h1: LET v=v1
170 IF h<=100 THEN GOTO 60
180 GOSUB 590
190 GOTO 220
200 LET h=0: LET c=0: GOSUB 390
210 GOSUB 650
220 STOP
250 LET h=100: LET f=100: LET t=0
260 LET v=5+FNr(10)
270 LET G=(FNr(40)+40)/100
280 RETURN
300 PLOT 180,8
310 FOR x=1 TO 15
320 DRAW 5,FNr(3)-2
330 NEXT x
340 PRINT "Gravity=";g
350 PRINT ''"Fuel:"'''''"Vel:"
360 PRINT ''''"Height:"'''''''"Thrust:"
370 RETURN
390 INVERSE c
400 LET y=h*1.3+10
410 PLOT 200,y: DRAW 34,0
420 DRAW -4,20: DRAW -13,10
430 DRAW -13,-10: DRAW -4,-20
440 RETURN
460 LET y=172-a*32
470 INK c
480 PLOT 0,y
490 DRAW b,0
500 DRAW INVERSE 1,100-b,0
510 RETURN
530 LET i$=INKEY$
540 IF i$="a" THEN LET t=t+4 :
    IF t>100 THEN LET t=100
550 IF i$="d" THEN LET t=t-4 :
    IF t<0 THEN LET t=0
560 IF t>f THEN LET t=f
570 RETURN
590 CLS
600 FOR i=1 TO 20
610 PRINT AT FNr(21),FNr(31);"*"
620 NEXT i
630 PRINT "Lost in space!!!"
640 RETURN
650 PRINT AT 0,0;"Landed at ";
    INT((v+v1)*5)/10'
660 IF (v+v1)<8 THEN GOTO 680
670 PRINT "All dead": RETURN
680 PRINT "Safely": RETURN
```

Touchdown: BBC version

```
20 MODE 5
30 *FX 12,1
40 PROCSETVAR
50 PROCDISPLAY
60 PROCMODULE(H,3)
70 PROCBAR(1,F,3+2*(F<25))
80 PROCBAR(2,ABSV,3+(V<0))
90 PROCBAR(3,H,3+2*(H<25))
100 PROCBAR(4,J,3)
110 PROCTHRUST
120 V1=V-T/20+G : F=F-T/10
130 H1=H-(V+V1)/10
140 PROCMODULE(H,0)
150 IF H1<0 THEN 200
160 H=H1:V=V1
170 IF H<=100 THEN 60
180 PROCLOST
190 GOTO 220
200 PROCMODULE(0,2)
210 PROCLANDED
220 *FX 12
230 END
240 DEF PROCSETVAR
250 H=100:F=100:T=0
260 V=5+RND(10)
270 G=(RND(40)+40)/100
288 ENDPROC
290 DEF PROCDISPLAY
300 MOVE 800,30
310 FOR X=800 TO 1280 STEP 16
320 DRAW X,10+RND(40)
330 NEXT
340 PRINT "GRAVITY=";G
350 PRINT ''''"FUEL:"'''''''"VEL:"
360 PRINT '''''"HEIGHT:"'''''''"THRUST:"''
370 ENDPROC
380 DEF PROCMODULE(H,C)
390 GCOL 0,C
400 Y=H*8.5+150
410 MOVE 1040,Y : PLOT 1,-40,-40
420 PLOT 1,-8,-60 : PLOT 1,96,0
430 PLOT 1,-8,60 : PLOT 1,-40,40
440 ENDPROC
450 DEF PROCBAR(N,V,C)
460 Y=1000-192*N
470 GCOL 0,C
480 MOVE 0,Y : MOVE 0,Y-16
490 PLOT 85,V*4,Y : PLOT 85,V*4,Y-16
500 PLOT 87,400,Y : PLOT 87,400,Y-16
510 ENDPROC
520 DEF PROCTHRUST
530 *FX 15,1
540 IF INKEY(-194) THEN T=T+4 :
    IF T>100 THEN T=100
550 IF INKEY(-179) THEN T=T-4 :
    IF T<0 THEN T=0
560 IF T>F THEN T=F
570 ENDPROC
580 DEF PROCLOST
590 CLS
600 FOR I=1 TO 20
610 VDU 31,RND(19),RND(31),42
620 NEXT
630 PRINT TAB(4,16)"LOST IN SPACE!!"
640 ENDPROC
650 DEF PROCLANDED
660 VDU 28,0,31,11,0,12
670 PRINT '"LANDED"
680 PRINT '"AT VEL ";INT((V+V1)*5)/10
690 IF (V+V1)<8 THEN PRINT '"SAFELY"
    ELSE PRINT '"ALL DEAD"
700 ENDPROC
```

Touchdown: Apple version

```
15 HOME
20 HGR
30 DEF FNR(X)=INT(RND(1)*X+1)
40 GOSUB 250
50 GOSUB 300
60 C=3:GOSUB 390
70 A=1:B=F:GOSUB 460
80 A=2:B=ABS(V):GOSUB 460
90 A=3:B=H:GOSUB 460
100 A=4:B=T:GOSUB 460
110 GOSUB 530
120 V1=V-T/20+G : F=F-T/10
130 H1=H-(V+V1)/10
140 C=0:GOSUB 390
150 IF H1<0 THEN 200
160 H=H1:V=V1
170 IF H<=100 THEN 60
180 GOSUB 590
190 GOTO 220
200 H=0:C=3:GOSUB 390
210 GOSUB 660
220 END
250 H=100:F=100:T=0
260 V=5+FNR(10)
270 G=(FNR(40)+40)/100
280 RETURN
300 HCOLOR=3
305 HPLOT 0,155
310 FOR X=0 TO 279 STEP 5
320 HPLOT TO X,159-FNR(10)
330 NEXT
335 FOR I=1 TO 30 :HPLOT FNR(279),
    FNR(150)
337 NEXT
340 VTAB 21 : PRINT TAB(34);"G=";G
350 VTAB 21 : PRINT "FUEL:" :
    PRINT "VEL:"
360 PRINT "HEIGHT:" : PRINT "THRUST:";
370 RETURN
390 HCOLOR=C
400 Y=(100-H)*1.3
410 HPLOT 140,Y TO 120,Y+10
420 HPLOT TO 120,Y+20 :
    HPLOT TO 160,Y+20
430 HPLOT TO 160,Y+10 : HPLOT TO 140,Y
435 HPLOT 155,Y+20 TO 160,Y+25
437 HPLOT 125,Y+20 TO 120,Y+25
440 RETURN
460 VTAB (20+A) : HTAB 8
470 INVERSE
480 PRINT SPC(B/4);
490 NORMAL
500 PRINT SPC(26-B/4);
510 RETURN
530 I$="" : IF PEEK(-16384)>127
    THEN GET I$
540 IF I$="A" THEN T=T+4 :
    IF T>100 THEN T=100
550 IF I$="D" THEN T=T-4 :
    IF T<0 THEN T=0
560 IF T>F THEN T=F
570 RETURN
590 HOME : VTAB 23
600 PRINT "LOST IN SPACE!!"
640 RETURN
660 HOME : VTAB 22
670 PRINT "LANDED AT VEL ";
    INT((V+V1)*5)/10
680 IF (V+V1)<8 THEN 700
690 PRINT "ALL DEAD" : RETURN
700 PRINT "SAFELY" : RETURN
```

131

Adding to the programs

Here are some ideas for additions you can make to the programs in this book or to your own programs. In most cases you won't be able to add these to a ZX81 with only 1K as the games themselves fill almost all its memory space, but you should find there is plenty of room on the other computers.

Remember you will either have to restrict your additions to the spare line numbers in a program or renumber the program. If you decide to renumber, take care you change all the GOTO and GOSUB lines too.

Getting the computer to tell you how to play

You can add a section to any program to make the computer print instructions telling you what to do. The easiest way to do this is to add some lines, such as those below, at the beginning of the program and then put a sub-routine at the end.
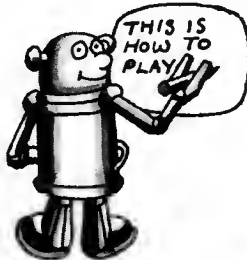
```
10 PRINT "TITLE OF GAME"
11 PRINT "DO YOU WANT TO"
12 PRINT "KNOW HOW TO PLAY?"
15 INPUT I$
S ZX17 IF I$(1)="Y" THEN GOSUB 1000
★■▲●17 IF LEFT$(I$,1)="Y" THEN GOSUB 1000
```

main program goes here

```
1000 PRINT "WHAT YOU HAVE TO"
1010 PRINT "DO IS........"
1999 RETURN
```

You can add as many print statements as you like for the instructions, just remember to put a number and the word PRINT at the beginning of each one. Restrict the length of the part inside the quotation marks to the number of characters your computer can print on one line. Don't forget to put a RETURN line at the end or the program won't work.

Making the computer stop and wait for you

If your instructions are very long, you may want to insert this sub-routine which stops the program running at a particular point until you press a key. This way you can stop the instructions scrolling off the top of the screen before you have read them. Put a GOSUB line at the place you want the program to stop and then put this sub-routine at the end.

```
        1000 PRINT "PRESS A KEY TO CONTINUE ";
■S ZX1010 IF INKEY$="" THEN GOTO 1010
★1010 I$=GET$
●1010 GET I$
▲1010 GET I$ : IF I$="" THEN GOTO 1010
        1020 PRINT
        1030 RETURN
```

Making the computer "talk" to you
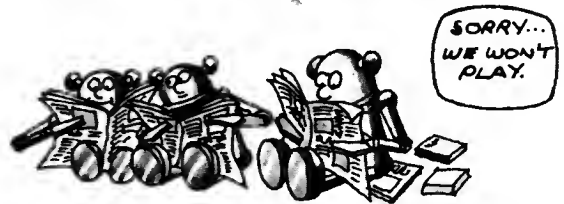
You can make the computer ask you questions and react to your answers. For instance, here is an addition which will make the computer refuse to play with you unless your name begins with J.

```
1 PRINT "WHAT IS YOUR NAME?"
2 INPUT I$
3 IF I$(1)<>"J" THEN GOTO 1000
3 IF LEFT$(I$,1)<>"J" THEN GOTO 1000
4 PRINT "OK-YOU CAN PLAY."
5 PRINT "ARE YOU READY?"
6 INPUT J$
S ZX7 IF J$(1)<>"Y" THEN GOTO 5
★■▲●7 IF LEFT$(J$,1)<>"Y" THEN GOTO 5
```

main program here

```
1000 PRINT "SORRY THIS GAME IS"
1010 PRINT "ONLY FOR PEOPLE"
1020 PRINT "WHOSE NAMES BEGIN"
1030 PRINT "WITH J"
```

Here is another one where the computer dares you to be brave enough to play.

```
 10 PRINT "VERY SCAREY GAME"
 12 PRINT "ARE YOU BRAVE ENOUGH"
 14 PRINT "TO TACKLE THE GREEN"
 15 PRINT "HAIRY MONSTER?"
 16 INPUT I$
S ZX17 IF I$(1)="Y" THEN GOTO 20
★■▲●17 IF LEFT$(I$,1)="Y" THEN GOTO 20
 18 PRINT "COWARD"
 19 STOP
```

You could combine this with the instruction sub-routine by taking lines 11 to 17 from the instructions section on this page and putting them at lines 20 to 26 of this program. You can then start the main program at line 30 and add the instruction sub-routine at the end.

Would you like another go?

Instead of typing RUN each time you play a game, you can make the computer ask you if you'd like another go. Put these lines at the end of the program, just before the last STOP statement.

```
1000 PRINT "DO YOU WANT ANOTHER GO?"
1010 INPUT I$
S ZX1020 IF I$(1)="Y" THEN RUN
★■▲●1020 IF LEFT$(I$,1)="Y" THEN RUN
1030 PRINT "OK THEN - BYE"
1040 STOP
```

Change line numbers according to your program.
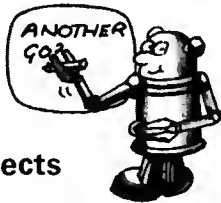
Adding sound effects

The BBC, VIC 20, ZX Spectrum and some Apples are able to produce sounds and you can add lines to your programs to make them do so at appropriate places. You could add an explosion for instance, or a little tune which plays if you win. All the computers need different instructions to make sounds though, so you will have to look at your manual. In some cases you can add a single line to your program at the place you want the sound. In others, you need several lines and it is best to put these in as a sub-routine.

As an example, here is the sound of a shot for the BBC. You can experiment with where to put it in the program, but you must give it a line number to make it work:

```
SOUND 0,-15,5,10
```

At the back of the VIC manual you will find some useful sub-routines for sounds such as "laser beam", "explosion" and "red alert". Put a GOSUB line where you want the sound to appear, number the sub-routine and add a RETURN at the end of it.

Special note for BBC and Spectrum users

If you have a BBC or a ZX Spectrum you may find that some of the games in this book run too fast for you. You will find a box next to these games containing instructions for changing the speed. Remember, to slow the game up you always need to use a higher number. Later models of the BBC may run up to twice as fast as the earlier models, and this could make the games appear impossible on the first run. Be prepared to make big changes to the speed number to correct this.

Writing your own games programs

As you work through the games in the book, you will probably find yourself making more and more changes to them and eventually wanting to write new games of your own. On these two pages you will find some hints on how to set about doing this.

Before you start, it is a good idea to stop and think about what your computer can and cannot do.

*It can ask you for information.
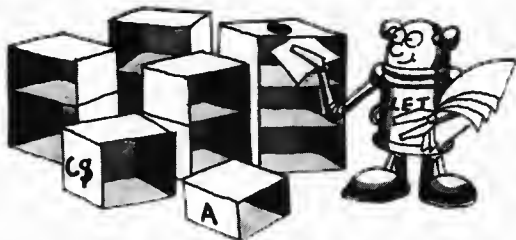
*It can store information

*It can select numbers at random by using RND.
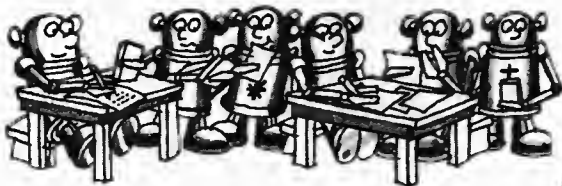
*It can do calculations.

*It cannot do anything unless you tell it to.
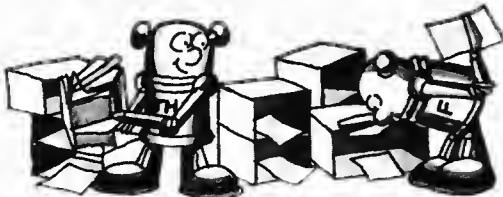
*It can make decisions by comparing items of information in various ways.

*Provided you use its language correctly, it can do only *exactly* what you tell it, even if it is silly.

Remember, when you are trying to work out a game, not to include anything which your computer won't be able to do.
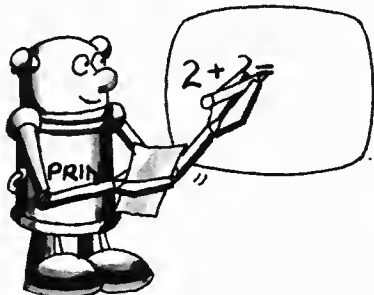
*It can tell you the results of its calculations and decisions and also what is stored in its memory.

Planning a game

Before you can tell the computer how to play your game, you must know exactly how to play it and what the rules are yourself. The computer will need a series of simple logical instructions, so work out your game in your head or on paper first and then break it down into simple steps.

Next write a plan (in English – don't try to use BASIC yet) of all the stages of the game in order.

Here is a plan for a simple shooting game, such as firing cannon balls at a pirate ship or shooting laser beams at an alien invader, to give you an idea.

PLAN

1) PRINT TITLE AND INSTRUCTIONS

2) CHOOSE A TARGET FOR THIS GAME

3) BEGIN A LOOP TO GIVE THE PLAYER N GOES

4) GET A SHOT FROM THE PLAYER

5) CHECK IF SHOT WAS ON TARGET

6) PRINT MESSAGE DEPENDING ON ACCURACY OF SHOT

7) GO BACK FOR ANOTHER GO IF SHOT WAS UNSUCCESSFUL

Writing the program

The next stage is to convert your plan into BASIC. Each step in your plan may need several lines in BASIC. Don't forget to leave gaps when numbering your program lines so you can go back and add extra ones if you need to.

Do a first draft of the program on paper first and then start testing on the computer. Your computer will spot errors much more quickly than you will see them yourself and may give you a clue as to what is wrong. Remember that debugging programs is a long, tedious process even for expert programmers, so don't expect to get yours right first time.

Once you have got the core of the program working, you can add to it. Scoring, extra comments, more targets etc. can all be incorporated later. You could add sections from the programs in this book to your games.

Don't expect to be able to write exciting and original games straight away. Keep your ideas very simple and be prepared to adapt them as you go along. You may find you have included something in your game which is easy for humans to do but very difficult for a computer. As you get more experienced you will begin to know instinctively what your computer can do and find it easier to write programs for it.

Summary of BASIC

This section lists some common BASIC words and describes what they make the computer do and how they are used. Most of them have been used in the programs in this book, so you can check back through the book to see how they work in a game. Not all the words can be used on all the computers mentioned in this book. The conversion chart on page 96 shows what you can use instead.
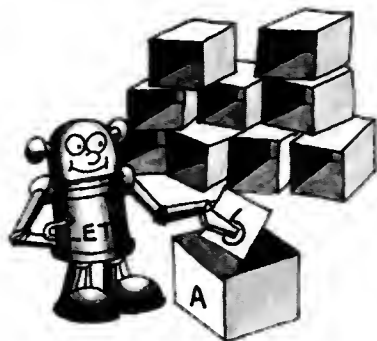
LET tells the computer to label a section of its memory and put a particular value in it e.g. LET A=6 means label a section of memory "A" and put the value 6 in it. "A" is called a "variable" and putting something in it is called "assigning a value to a variable".

Some variable labels are followed by a dollar sign e.g. A$. This means they are for "strings", which can contain any number of characters, including letters, numbers and symbols.
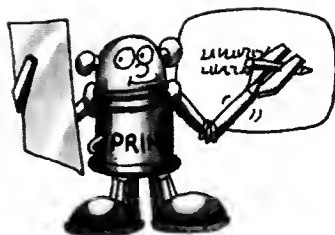
PRINT tells the computer to display things on the screen and you can use it in several ways:

A message enclosed in quotation marks with PRINT in front of it will be displayed on the screen exactly as you typed it. The section inside quotes does not have to be in BASIC, it can be anything you like.
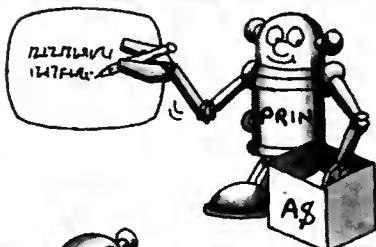
PRINT followed by a variable label e.g. PRINT A or PRINT A$ tells the computer to display the contents of that variable on the screen.

PRINT can also do calculations and then display the results e.g. PRINT 6*4 will make the computer display 24.

You can use PRINT by itself to leave an empty line.
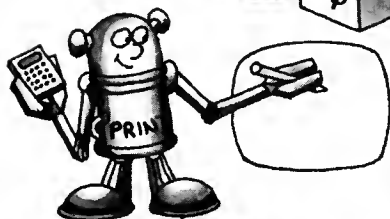
RND tells the computer to choose a number at random. Different computers use different forms of RND and you can see what these are in the conversion chart on page 96. On Sinclair computers RND by itself produces a number between 0 and 0.99999999. You can vary the limits of the number it chooses by multiplying RND and adding to it. E.g. RND*20 produces a number between 0 and 19.99999999, while RND*20+1 produces a number between 1 and 20.99999999.

See INT for how to produce only whole numbers.

See CHR$ for how to produce letters and other keyboard characters at random.
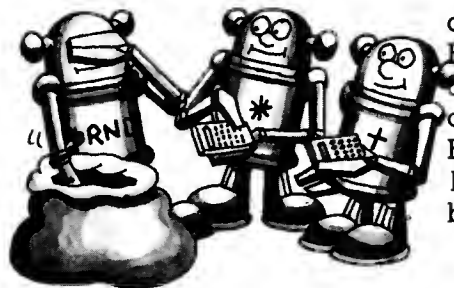
INT is short for integer, which means whole number. For positive numbers, it tells the computer to ignore everything to the right of the decimal point. E.g. INT(20.999) is 20. For negative numbers, it ignores everything to the right of the decimal point and "increases" the number to the left of it by one e.g. INT(−3.6) is −4.

INT is often used with RND, like this: INT(RND*20+1) which tells the computer you want it to choose a whole number between 1 and 20.
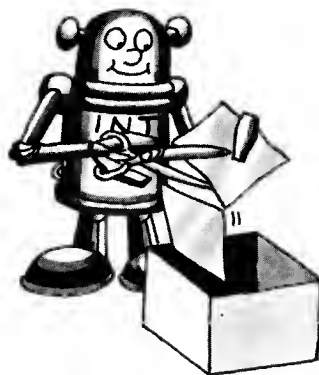
**CHR$** converts numbers into letters. Apart from the ZX81, all the computers in this book use the ASCII*set of keyboard characters in which each character corresponds to a certain number. E.g. letter A has the code number 65 and PRINT CHR$(65) will display an A on the screen.

You can use CHR$ with INT and RND to make the computer select random letters, like this:

CHR$(INT(RND*26+65))

This line will produce random letters on a ZX Spectrum (see conversion chart for other computers).
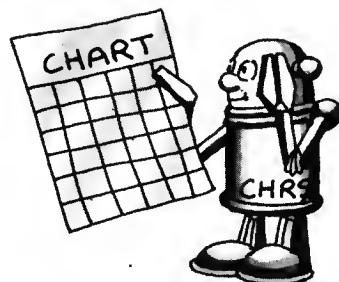
FOR is used to start a "loop" which will make the computer repeat part of a program a certain number of times. It must be followed by a variable (such as G to stand for the number of goes allowed in a game), and the variable must be given start and end values (such as 1 TO 10.)

The end of the loop is marked by a NEXT line (NEXT G in this example) which increases the value of the variable by 1 each time and then sends the computer back to the FOR line again. When the variable reaches its end value, the computer ignores the NEXT line and carries on to the line which follows it. Every FOR must have a NEXT or you will get a bug.
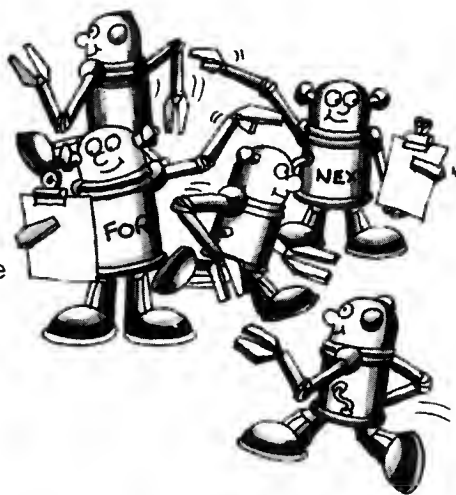
INPUT labels a space in the computer's memory, prints a question mark and then waits for you to type something which it can put in this memory space. It will not carry on with the rest of the program until you press RETURN, ENTER or NEWLINE.

You can use number or string variables with INPUT, but if you use a number variable the computer will not accept letters from you.

137
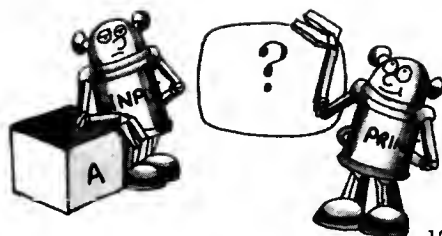
*American Standard Code for Information Interchange (see page 95)
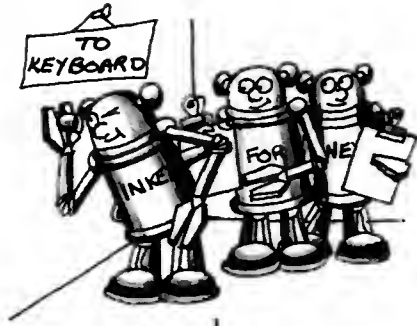
INKEY$ checks the keyboard to see if a key is being pressed and if so which one. It does not wait for you to press a key like INPUT does. It is usually used in a loop which makes the computer go round checking the keyboard lots of times. This is because computers work so quickly, you wouldn't have a chance of pressing a key in the time it takes the computer to do one check.

If you haven't pressed a key before the loop finishes, the computer carries on with a string containing nothing (called a "null" string). NB Apple and VIC do not use INKEY$.
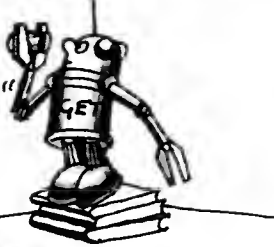
GET is used instead of INKEY$ on VIC and Pet computers.
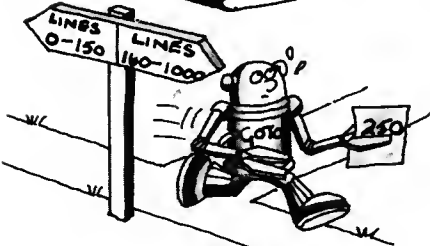
GOTO makes the computer jump up or down the program ignoring the lines in between. You must put the number of the line you want it to jump to after the GOTO instruction.
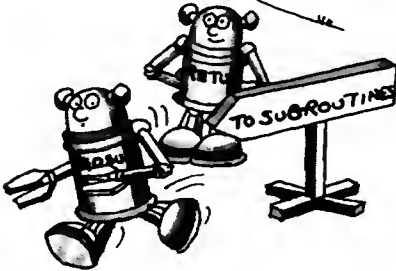
GOSUB tells the computer to leave the main program and go to a sub-routine. GOSUB must be folowed by the number of the first line of the sub-routine. At the end of the sub-routine you must have a RETURN line. This sends the computer back to the main program to the line immediately following the GOSUB line. A GOSUB without a RETURN in a program will give a bug.
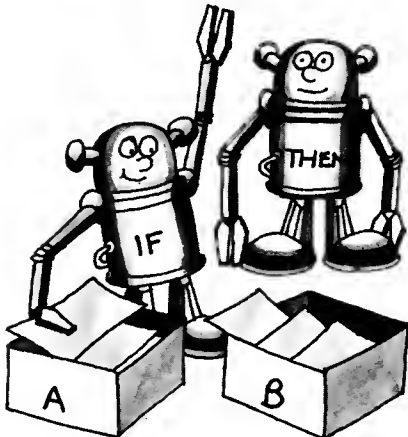
IF ... THEN tells the computer to decide if an expression is true or false, and do different things depending on the answer. It is used with the following signs, and also with AND or OR:

=the same as
<less than
>greater than
<=less than or the same as
>=greater than or the same as
<>not the same as

If the computer decides an expression is true, it carries on to do the instruction which follows THEN. If it decides it is false, it ignores the rest of that line and goes on to the next one.

138

CLS is used to clear everything off the screen without removing or changing anything in the memory. It is useful for removing the listing from the screen at the beginning of a RUN or in games when you want the player to react to something seen for a limited amount of time. (NB Apple and VIC do not use CLS – see conversion chart).
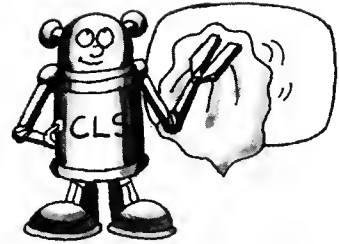
HOME is used by Apple computers instead of CLS to clear the screen.
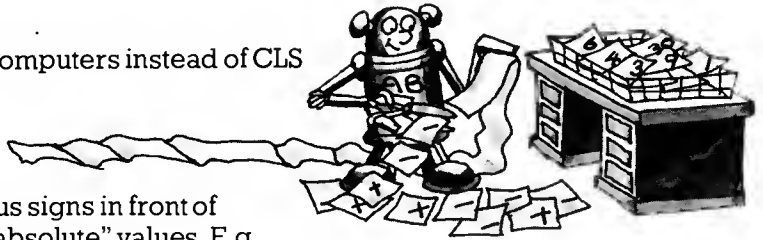
ABS ignores plus and minus signs in front of numbers and takes their "absolute" values. E.g. ABS(−10) is 10 and ABS(+10) is also 10.

VAL takes the numeric value of numbers written as strings. In effect, it tells the computer to ignore the dollar sign and treat the string as an ordinary number variable. E.g. if I\$ = "60" then VAL(I\$) is the number 60.
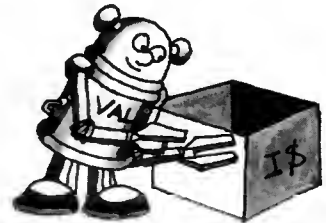
ASC converts a character into its ASCII code number e.g. ASC("3") gives 51. The expression in brackets must be a string e.g. ASC(A\$) or ASC("20").
NB ZX81 and ZX Spectrum do not use ASC, though the Spectrum does use the ASCII code.
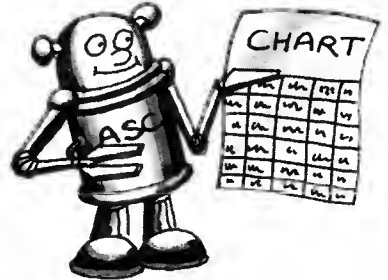
CODE is used by ZX81 and Spectrum in place of ASC. Like ASC it must always be followed by a string. Remember that the ZX81 uses different code numbers from the other computers.

TAB moves the cursor across the screen to a specified column number. It is usually used with PRINT to display something in the middle of the screen. The number of spaces you want the cursor moved is put in brackets after TAB. The maximum number you can use depends on the screen width of your computer.

139
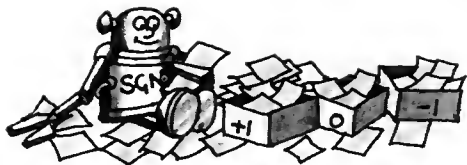
SGN tells the computer to find out the sign of a number. It produces − 1 for a negative number, 0 for zero and + 1 for positive numbers. E.g. SGN(−30) is − 1, SGN(7) is + 1 and SGN(0) is 0.
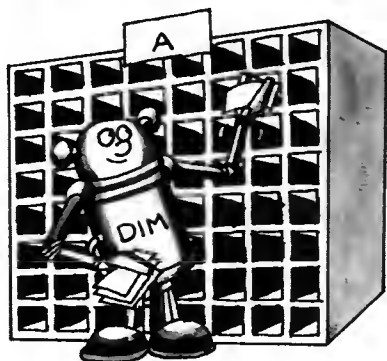
DIM tells the computer how much memory space will be needed for an "array" (a row or a grid). E.g. DIM X(6) tells the computer to set aside an area large enough to contain a row of 6 elements and labelled X. DIM A(8,8) means a memory space labelled A and big enough to take 8 elements across and 8 down is needed. The number of elements of data used in the program must correspond to the numbers in brackets after DIM or you will get a bug.

SQR takes square roots of numbers. E.g. SQR(16) gives the answer 4.

SIN calculates the sine of an angle. In a right-angled triangle the length of the side opposite an angle, divided by the length of the hypotenuse (the side opposite the right angle) is the sine of that angle. When you use SIN in a program, the angle you are using it with must be measured in radians, not degrees.
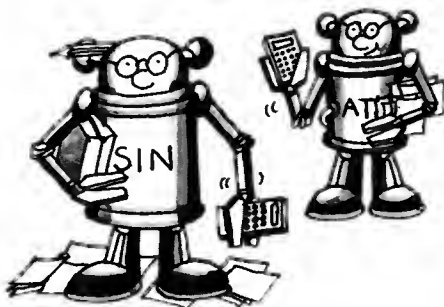
ATN is one of the trig. functions which computers can calculate (see also SIN above). It stands for arctangent and it is important to remember that it gives an answer in radians, not degrees. You will need to use a maths book to find out how this works if you do not already know about it.
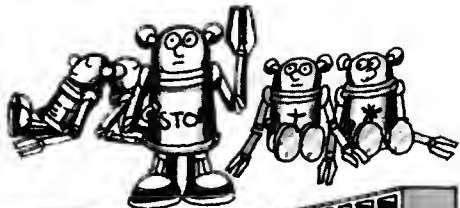
STOP tells the computer not to go any further in a program. Computers other than the ZX81 can use END instead.

PEEK is a way of finding out what is in a specific area of the computer's memory. You need to use it with a number which specifies an "address" in the memory.
NB not used on BBC.

POKE is a special way of putting information in the computer's memory by using a memory "address".
NB not used on BBC.

Answers

You may find that your answers to some of the puzzles are different to the ones given here. As long as they work on your computer then this doesn't really matter, but check to see if they are as neat and simple as the answers in the book.

Page 101
Starship Takeoff

Lines 30 and 40 select the numbers which determine what the force will be. To increase the range of possible forces, you can increase either the 20 in line 30, or the 40 in line 40, or both of these numbers. Increasing the range of forces will obviously make the game more difficult.

Page 103
Intergalactic Games

Change lines 222 and 230 as follows:

```
222 LET B=B+INT(1000/G)
230 GOTO 20
```

and add a new line 15:

```
15 LET B=0
```

Page 105
Evil Alien

Change lines 20 and 30 and add a new line 25 as follows:

```
20 PRINT "HOW DIFFICULT? (6 TO 30)"
25 INPUT S
30 LET G=INT(S/3)
```

Page 107
Beat the Bug Eyes

To make the bugs appear in more than four places on the screen, you need to put a higher number than 4 in the middle of line 70, change line 80 and add more sub-routines at the end of the program – one for each extra position.

Here are the changes to make the bugs appear in 5 places:

```
70 LET R=INT(RND*5+1)
```

```
S ZX80 GOSUB 220+20*R
★■▲●80 ON R GOSUB 240,260,280,300,320
```

```
240 LET D=5
245 LET A=1
250 GOTO 350
260 LET D=1
265 LET A=9
270 GOTO 350
280 LET D=5
285 LET A=18
290 GOTO 350
300 LET D=10
305 LET A=7
310 GOTO 350
320 LET D=15
325 LET A=15
330 GOTO 350
```

> You can use any numbers you like for A and D provided they will fit on your screen.

To add more bugs, change the 10 in lines 30 and 220 to a higher number. (Make sure you use the same number for both lines.)

Page 109
Moonlander

To increase the speed allowed for a safe landing, you need to make changes to lines 230, 240 and 250. You can use any numbers you like – the higher they are the easier the game will be. In this example, you are allowed a speed of 2 for a good landing and 7 for an OK landing:

```
230 IF V1>7 THEN PRINT "YOU CRASHED
       - ALL DEAD"
240 IF V1>2 AND V1<=7 THEN PRINT "OK
       -BUT SOME INJURIES"
250 IF V1<=2 THEN PRINT "GOOD LANDING"
```

Page 111
Monsters of Galacticon

Four ways of making this game harder are:

1 Start the game with less people in the group by putting a smaller number than 5 in line 40.

2 Increase the number of monsters by changing the 4 in lines 20 and 30. Add

141

extra monster names at lines 81 to 89 using M$(5) and M$(6).

3 Reduce the number of goes allowed by altering the 8 in line 160.

4 Increase the chance of the monster being angered in line 330 by increasing .4 a little.

Page 113
Alien Snipers
In this game, N is the code number. To change the scoring to fit the code number you need to increase the score by N each time instead of 1. So, change line 190 as follows:

```
S ZX190 IF I$=CHR$(CODE(L$)+N) THEN
            LET S=S+N
★■▲●190 IF I$=CHR$(ASC(L$)+N) THEN
            LET S=S+N
```

Page 115
Asteroid Belt

You need to change line 260 so that the computer adds the number of stars to your score instead of 1. The number of stars is controlled by the value chosen for N in line 70, so, as in the puzzle above, you need to add N to the score. You also need to change line 320.

```
260 LET S=S+N
320 PRINT "YOU SCORED ";S;" POINTS"
```

Page 117
Trip into the Future

1 To increase the range of years which must elapse before you return to Earth, change the 100 in line 30 to a higher number, e.g. 150, like this:

```
30 LET T=INT(RND*150+25)
```

2 To increase the accuracy from 5 to 2 years, change the 5s in lines 180 and 190 to 2, like this:

```
180 IF ABS(T-T2)<=2 THEN PRINT "YOU
        ARRIVED ON TIME"
190 IF ABS(T-T2)>2 THEN PRINT "NOT
        EVEN CLOSE"
```

3 Line 170 contains the number which determines the length of your lifetime. Change the 50 to a higher number for a longer lifetime.

Page 119
Death Valley

You can make the valley longer by changing the number in line 30 to something higher than 200.

Page 121
Space Mines

Add these lines to make the computer ask if you would like another game:

```
645 PRINT "ANOTHER GAME? (TYPE Y OR N)"
646 INPUT A$
647 IF A$="Y" THEN GOTO 10
```

You must then a new line at 5 and change line 30 to add the money you ended up with at the end of the game to the money allowed for the new game:

```
5 LET M=0
30 LET M=M+INT(RND*50+10)*P
```

(Make sure you use the correct version of RND for your computer.)

Index

THE BEGINNER'S
COMPUTER HANDBOOK
Understanding & programming the micro

This is a colourful introduction to the world of microcomputers for beginners
of all ages. It begins by explaining how computers work, what they can do and
why you might want to own one, and includes a buyer's guide which lists the
main features of most of the home computers currently available. Then
follows a step-by-step guide to programming in BASIC, the language used by
most microcomputers. This section includes problems to solve and programs
which will run on any microcomputer. At the back of the book there are 13
program listings* for spacegames. Explanations of how the programs work,
suggestions for changing them and puzzles are given alongside the listings,
and there are hints on writing your own games programs. With a look-up
guide to BASIC, a BASIC conversion chart and a glossary of computer jargon,
this book will be an invaluable guide to anyone who wants to know about
microcomputers.

*The spacegames programs will work on the following computers: BBC,
ZX Spectrum, ZX81, VIC 20, Pet, Apple and TRS-80.

The material in this book is also available in three separate books with titles:
Understanding the Micro, Introduction to Computer Programming and
Computer Spacegames.

COMPUTER HANDBOOK

MICROS • PROGRAMMING
GAMES PROGRAMS

🪂 USBORNE